EXHIBIT 1

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF MASSACHUSETTS

SINGULAR COMPUTING LLC, Plaintiff, v. GOOGLE LLC, Defendant.	C.A. No. 1:19-cv-12551-FDS

EXPERT REPORT OF MIRIAM LEESER, PH.D. REGARDING INVALIDITY

Executed on 12/20/2022

DocuSigned by:
Miriam Leeser, Ph.D.
A3ED90A1A36A4F5...

Miriam Leeser, Ph.D.

Table of Contents

I. INTRODUCTION

1. I understand that Plaintiff Singular Computing LLC ("Singular") has filed a patent infringement lawsuit against Defendant Google LLC ("Google"), captioned *Singular Computing LLC v. Google LLC*, Case No. 1:19-cv-12551-FDS, in the U.S. District Court for the District of Massachusetts, in which Singular alleges that Google infringes certain claims of U.S. patents issued to Singular. Specifically, I understand that Singular asserts that Google infringes the following claims of two patents: claim 53 of U.S. Patent No. 8,407,273 ("the '273 patent") and claim 7 of U.S. Patent No. 9,218,156 ("the '156 patent") (the "Patents-in-Suit" or "Asserted Patents"). I further understand that Google has asserted that these two patent claims (the "Asserted Claims") are invalid as anticipated and/or obvious in light of certain prior art.

2. I have been retained by Google to evaluate and offer my opinions regarding whether the Asserted Claims are invalid as anticipated and/or obvious. Specifically, I have been asked to evaluate and offer my opinions on whether a particular system that I was involved in developing (which I describe in further detail below) anticipates and/or renders obvious the Asserted Claims. Given my personal involvement in the development, use, and public disclosure of and regarding this system (as described in more detail below), I have personal knowledge of many of the facts set forth in this report, and am a percipient witness as to those facts, in addition to offering the opinions set forth in this report.

3. I am being compensated for my time on this case at the rate of $750 per hour, and my compensation does not depend on the opinions expressed here, on any testimony that I may give, or on the outcome of this case.

4. I have reviewed the Asserted Patents, their prosecution histories, and the Court's July 27, 2022 Memorandum and Order on Claim Construction in forming the opinions expressed in this report. I have also reviewed materials regarding my prior work developing a library of hardware modules for performing floating-point arithmetic on a variety of custom floating-point formats using reconfigurable hardware, which was deployed to hardware comprising multiple

field-programmable gate arrays (FPGAs). A list of the materials that I have considered in forming my opinions is attached as **Exhibit A**.

5. I reserve the right to supplement my opinions in response to any future ruling by the Court that impacts the scope or meaning of the Asserted Claims. I also reserve the right to supplement my opinions in response to new contentions and/or positions that Singular may advance prior to or during trial.

II. BACKGROUND AND QUALIFICATIONS

6. My name is Miriam Leeser. I am a professor of Electrical and Computer Engineering at the Northeastern University College of Engineering, where I have been since 1996.

7. Since receiving my Ph.D. in Computer Science from Cambridge University in 1988, the focus of my research has covered a wide range of subjects generally related to digital hardware design, FPGAs, computer arithmetic and image processing, and I have developed particular expertise in custom floating point designs.

8. My research focus at Northeastern is accelerators for computer-intensive applications from the edge to the cloud, including applications of FPGAs for wireless communications, machine learning, image processing, and data privacy. My research has been supported by both the public and private sector, including governmental agencies such as the National Science Foundation (NSF) and the Defense Advanced Research Projects Agency (DARPA), as well as companies such as Microsoft, Google, Mathworks, Intel, Digital Equipment Corp., Samsung, AMD, and others.

9. I am currently the head of the Reconfigurable and GPU Computing Laboratory (RCL) at the Northeastern University College of Engineering (formerly known as the Reconfigurable Computing Laboratory, and before that, the Rapid Prototyping Laboratory), a position I have held since 1996. At the RCL we research how to use hardware accelerators such as Field Programmable Gate Arrays (FPGAs) and Graphics Processing Units (GPUs) for a wide

variety of applications. The focus is in developing tools, libraries, and interfaces that make accelerators easier to use. As just a few examples, our work in the RCL has helped speed up wireless networking, Internet of Things (IoT) technology, medical imaging, and security and privacy applications, in many cases incorporating theoretical algorithm advances and making them practically viable for the first time.

10. At Northeastern, I also currently serve as a member of the Computer Engineering faculty group, which offers courses and leads research projects in a variety of topics related to computer engineering, including hardware and software design, computer architecture, algorithms, implementations and applications.

11. I obtained a Bachelor of Science degree in Electrical Engineering, with distinction, from Cornell University in 1980, and a Ph.D. in Computer Science from Cambridge University in 1988.

12. Prior to my Ph.D. studies, I worked as a hardware development engineer at Codex Corporation, a division of Motorola, where I worked on computer networking hardware.

13. I am and have been deeply involved in a variety of professional organizations related to my areas of research and expertise. For example, I am currently a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE), which is the world's largest technical professional organization. I have served in various leadership capacities for IEEE, including, for example: as General Chair of the 20th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP), which was held in 2009 in Boston, Massachusetts; as Publications Chair for the IEEE High Performance Extreme Computing Conference (HPEC) from 2012-2022; as Program Chair for the 21st IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM), which was held in 2013 in Seattle, Washington; as General Chair of the 22nd IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM), which was held in 2014 in Boston, Massachusetts; and on the Steering Committee for IEEE FCCM from 2013-

2019. I have also served on the Program Committee for a number of IEEE conferences, as detailed in my curriculum vitae.

14. I am also currently a Senior Member of the Association for Computing Machinery (ACM), which is the world's largest scientific and educational computing society. I have served in various leadership capacities for ACM, including, for example as Artifact Evaluation Chair for the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays from 2020 through 2023 and on the Program Committee for the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays from 2009-2023. I have also served as editor or guest editor for various ACM publications, including as Associate Editor for ACM Transactions on Reconfigurable Technology and Systems (TRETS), which is a peer-reviewed journal covering technology, systems, and applications on reconfigurable computers or devices. ACM TRETS is the world's most prestigious peer-reviewed publication on reconfigurable computing technology and has published many seminal papers in the field.

15. I have received a variety of professional distinctions over the course of my career. In March 2022, for example, I was recognized as a Charter Member of the IEEE Computer Society Distinguished Contributors Program. The Program was introduced in 2021 to showcase the combined technical expertise and innovation power of IEEE's membership as well as recognize the voluntary commitment of IEEE members such as myself. Relatively early in my career, in 1992, I received the NSF National Young Investigator Award, which is considered one of the highest honors that can be bestowed by the U.S. National Science Foundation, for work on developing a toolkit for the design of floating-point arithmetic hardware and software. I was also selected as a 2018-2019 U.S. Fulbright Scholar, as a result of which I spent much of that academic year abroad as a Visiting Scholar at Maynooth University in Ireland, collaborating with researchers there and at the CONNECT Centre at Trinity College, which is the Science Foundation Ireland Research Centre for Future Networks and Communications. I have also been invited to speak and work at various other academic institutions, including a secondary academic appointment as a Visiting Scientist from June to December 2002 at the Massachusetts Institute of

Technology's (MIT) Lincoln Laboratories, which is a federally funded research and development center affiliated with MIT and operated by the U.S. Department of Defense, and which was chartered to apply advanced technology to problems of national security.

16. I am an author or co-author of over 180 refereed publications, including 45 peer-reviewed published journal articles, as well as the book *Hardware Specification, Verification, and Synthesis: Mathematical Aspects*. I have been an invited speaker at over 70 conferences, symposia, and workshops on topics within my area of research and expertise. I have received extensive recognition for my published research in the areas of high performance computing and reconfigurable hardware, including various best paper awards; for example, for my paper "Accelerating Matrix Processing for MIMO Systems" with Jieming Xu at the International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies (HEART 2021), and my paper "FINN-R: An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks," which was published in ACM Transactions on Reconfigurable Technology and Systems (TRETS), Special Section on Deep Learning on FPGAs, Vol. 11, Issue 3 (December 2018), and received a best paper award from ACM TRETS in 2020.

17. I am also a named inventor on U.S. Patent No. 4,967,344, which is titled "Interconnection Network for Multiple Processors" and generally relates to techniques for connecting a large number of independent processors together using a packet-based network (data bus) to allow very high-throughput data transfer between the processors. This patent arose from my time working in industry (at Codex Corp.).

18. Additional information regarding my background, qualifications, and experience, including a list of publications, conference and symposium presentations, and talks, can be found in my *curriculum vitae*, which is attached as **Exhibit B**.

III. SUMMARY OF OPINIONS

19. For the reasons set forth in the body of this report, it is my opinion that claim 53 of the '273 patent is invalid as anticipated due to the claimed invention being known or used by

others in this country before the priority date of the Asserted Patents, being in public use in this country more than one year prior to the priority date for the Asserted Patents, and/or being previously made in this country by another inventor who had not abandoned, suppressed, or concealed the invention, based on the creation, public availability, public disclosure, public knowledge, and public use of and regarding VFLOAT and its use on FPGA hardware. In addition, it is my further opinion that, even if claim 53 of the '273 patent is not anticipated, the claimed invention would have been obvious to a person having ordinary skill in the art at the time based on the creation, public availability, public disclosure, public knowledge, and public use of and regarding VFLOAT in combination with FPGA hardware available at the time of the claimed invention.

20. Likewise, it is my opinion that claim 7 of the '156 patent is invalid as anticipated due to the claimed invention being known or used by others in this country before the priority date of the Asserted Patents, being in public use in this country more than one year prior to the priority date for the Asserted Patents, and/or being previously made in this country by another inventor who had not abandoned, suppressed, or concealed the invention, based on the creation, public availability, public disclosure, public knowledge, and public use of and regarding VFLOAT and its use on FPGA hardware. In addition, it is my further opinion that, even if claim 53 of the '273 patent is not anticipated, the claimed invention would have been obvious to a person having ordinary skill in the art at the time based on the creation, public availability, public disclosure, public knowledge, and public use of and regarding VFLOAT in combination with FPGA hardware available at the time of the claimed invention

IV. APPLICABLE LEGAL STANDARDS

21. I am not an attorney and have no formal legal training. I have been informed of the relevant legal standards that apply in evaluating the validity of a patent from Google's attorneys and am relying on those instructions for these legal standards. Below I describe my understanding of these legal standards.

A. Priority Date and Scope of Prior Art

22. I understand that Singular has contended that the Asserted Patents are entitled to a priority date of at least June 19, 2009, which is the filing date of provisional patent application No. 61/218,691,[1] because the Asserted Patents claim priority via a chain of continuation applications to that provisional application.

23. My opinions as set forth in this report regarding the invalidity of the Asserted Patents are based on knowledge and/or use, public use, and prior invention of a system that predates the priority date of the Asserted Patents.

B. Claim Construction

24. I have been instructed and understand that claim construction is a matter of law for the Court to decide. I further understand that the Court issued a Memorandum and Order on Claim Construction ("Claim Construction Order") on July 27, 2022, setting forth the construction of various claim terms in the Asserted Patents.[2] I have reviewed the Claim Construction Order and applied the Court's rulings in reaching my opinions.

25. Specifically, I understand that the Court construed the terms "low precision and high dynamic range," "execution unit," and "first input signal representing a numerical value."

26. First, I understand that the Court construed "low precision and high dynamic range" as defined elsewhere in the claim language itself.[3] Specifically, the Court's Claim Construction Order explained Singular's position that the term "low precision" requires no construction because the term is defined in the claim—in particular, by the following claim language,[4] which appears in both claim 36 of the '273 patent (from which asserted claim 53 of

[1] *See* '156 patent at [60].
[2] Dkt. 354, Mem. and Order on Claim Construction.
[3] Dkt. 354, Mem. and Order on Claim Construction at 16-17 ("Therefore, the Court will construe the term 'low precision and high dynamic range' as defined in the claim itself.").
[4] Dkt. 354, Mem. and Order on Claim Construction at 16-17 ("Singular contends that the term 'low precision and high dynamic range' requires no construction because the term is defined in the claim. . . . According to Singular, the term 'low precision' is defined by the following claim language: . . . .").

the '273 patent depends) and claim 1 of the '156 patent (from which asserted claim 7 of the '156 patent depends):

> for at least X=5% of the possible valid inputs to the first operation, the statistical mean, over repeated execution of the first operation on each specific input from the at least X% of the possible valid inputs to the first operation, of the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input differs by at least Y=0.05% from the result of an exact mathematical calculation of the first operation on the numerical values of that same input[5]

The Court's Claim Construction Order also explained Singular's position that the term "high dynamic range" is defined by the following claim language,[6] which appears in both Asserted Claims: "the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/1,000,000 through 1,000,000."[7] The Claim Construction Order concluded that "the Court will construe the term 'low precision and high dynamic range' as defined in the claim itself,"[8] which I understand to refer, respectively, to the above-quoted claim language reciting "for at least X=5% of the possible valid input . . ." and "the dynamic range of the possible valid inputs . . ."

27. I also understand that the Court construed "execution unit" to mean "processing element comprising an arithmetic circuit paired with a memory circuit."[9]

28. I further understand that the Court construed "first input signal representing a numerical value" according to its plain and ordinary meaning.[10] In so doing, the Court observed that "Google construes the claim language such that the LPHDR execution unit operates on numerical values, which are represented by electrical signals, whereas Singular interprets the

[5] Dkt. 354, Mem. and Order on Claim Construction at 16-17 (quoting claim 36 of the '273 patent).
[6] Dkt. 354, Mem. and Order on Claim Construction at 17 ("[Singular] further contends that the term 'high dynamic range' is defined by the claims as 'the dynamic range of possible valid inputs to the first operation is at least as wide from 1/1,000,000 through 1,000,000 . . . .'").
[7] Dkt. 354, Mem. and Order on Claim Construction at 17 (quoting asserted claim 53 of the '273 patent).
[8] Dkt. 354, Mem. and Order on Claim Construction at 17.
[9] Dkt. 354, Mem. and Order on Claim Construction at 25.
[10] Dkt. 354, Mem. and Order on Claim Construction at 30.

claim language such that the LPHDR execution unit operates on signals that represent numerical values. *In this context, at least, that appears to be a distinction without a meaningful difference.*" Accordingly, the Court ruled that "the distinction between a signal (which represents an abstract value) and a value (which exists in the physical form of an electrical signal) is meaningless in this context . . . ."[11] For purposes of my analysis, therefore, I generally treat signals and the particular numerical values represented by those signals as one and the same.

29. Finally, I understand that the Court addressed the parties' dispute over whether the claim term "repeated execution" was indefinite, reaching the conclusion that it did not find that this term was indefinite.[12]

30. For claim terms not submitted for construction, or for claim terms where I am otherwise applying their "plain meaning," I have given claim terms their ordinary and customary meaning within the context of the patent in which the terms are used, *i.e.*, the meaning that the term would have had to a person of ordinary skill in the art in question at the time of the alleged invention.

31. I also understand that extrinsic evidence, such as dictionaries and treatises or articles known to one skilled in the art, can be considered so long as they do not contradict the patent's claims, description, or prosecution history.

C. Dependent Claims

32. I understand that patents may contain both independent claims and dependent claims. I understand that a dependent claim is one that references another claim, which may be a dependent or independent claim (sometimes called the "base" claim), and which is interpreted to incorporate all the elements or limitations of the claim(s) from which it depends. For example, I understand that a dependent claim reciting "The device of claim *x*" followed by various limitations requires or incorporates all the limitations of claim *x* as well as the further limitations specified in the dependent claim.

[11] Dkt. 354, Mem. and Order on Claim Construction at 30.
[12] Dkt. 354, Mem. and Order on Claim Construction at 15-16.

D. Burden of Proof Regarding Patent Invalidity

33. I understand that patents are presumed valid and that invalidity must be shown by "clear and convincing" evidence. I also understand that the "clear and convincing" standard is higher than a preponderance of the evidence but lower than the standard in criminal cases of proof beyond a reasonable doubt. I further understand that clear and convincing evidence is evidence that produces an abiding conviction that the truth of a fact is highly probable. I understand that the determination of whether the burden of proof is met is typically decided by a jury, although it can also be decided by the Court if there are no disputed questions of fact.

E. Anticipation

34. I understand that, once the claims of a patent have been properly construed, the second step in determining the validity of a patent claim requires a comparison of the properly construed claim language to the prior art on a limitation-by-limitation basis.

35. I understand that a patent claim is invalid if the claimed invention is not new. For the claim to be invalid because it is not new, all of its requirements must have existed in a single device or method that predates the claimed invention, or must have been described in a single previous publication or patent that predates the claimed invention. In patent law, these previous devices, methods, publications or patents are called "prior art references." If a patent claim is not new we say it is "anticipated" by a prior art reference.

36. I understand that a patent claim is "anticipated" if each and every element of the claim, as properly construed, has been disclosed in a single prior art reference either expressly or inherently (*i.e.*, necessarily present even if not expressly stated), and the claimed arrangement or combination of those elements has been disclosed, either expressly or inherently, in the same prior art reference.

37. I also understand that prior art includes various categories of information such as printed publications, patents, actual commercial products, source code, and/or other physical embodiments. Furthermore, I understand that there are a variety of ways that to show that a patent claim was not new, including the following:

a. if the claimed invention was already publicly known or publicly used by others in the United States before the priority date;

b. if the claimed invention was already being used in the United States before the priority date and that use was not primarily an experimental use (a) controlled by the inventor, and (b) to test whether the invention worked for its intended purpose; and

c. if the claimed invention was already made by someone else in the United States before the priority date, if that other person had not abandoned the invention or kept it secret.

38. I further understand that the person who first conceived of the claimed invention and first reduced it to practice is the first inventor. Conception is the mental part of an inventive act and is proven when the invention is shown in its complete form by drawings, disclosure to another or other forms of evidence, whereas a claimed invention is "reduced to practice" when it has been tested sufficiently to show that it will work for its intended purpose or when it is fully described in a patent application filed with the PTO.

39. I understand that although anticipation cannot be established by combining references, additional references may be used to interpret the anticipating reference by, for example, providing background indicating what the anticipating reference would have meant to a person of ordinary skill in the art. In addition, the description provided in the prior art must be such that a person of ordinary skill in the art in the field of invention could, based on the reference, practice the invention without undue experimentation at the time of the alleged invention of the asserted patent.

F. Repetition or Duplication of Claim Elements

40. It is my understanding that, in general, the mere duplication or repetition of parts or elements of a claimed invention may, in certain circumstances, be deemed to have no significance in the patentability of the claim over the prior art when no new and unexpected

result is produced by the duplication or repetition. For example, I understand that merely increasing or decreasing the number or quantity of a specific component or element of a claimed invention to yield a different number or quantity as compared to the prior art will generally not be deemed a patentable distinction over the prior art unless the increase or decrease produces a new and unexpected result.

G. Level of Ordinary Skill

41. I have been asked to offer my opinion regarding the level of ordinary skill in the art with respect to the '156 and '273 patents.

42. I understand that a person of ordinary skill in the art is presumed to be one who thinks along the lines of conventional wisdom in the art, and is a person of ordinary creativity, not an automaton.

43. I understand that the hypothetical person of ordinary skill in the art for a particular patent is determined as of the time of the alleged invention. Here, the priority date claimed by Singular for the Patents-in-Suit is the date of the provisional patent application, June 19, 2009.

44. I further understand that the hypothetical person of ordinary skill in the art is presumed to have knowledge of all references that are sufficiently related to one another and to the pertinent art, and to have knowledge of all arts reasonably pertinent to the particular problem that the claimed invention addresses.

45. To assess the level of ordinary skill in the art, I understand one considers the type of problems encountered in the art, the prior solutions to those problems found in prior art references, the rapidity with which innovations are made, the sophistication of the technology, and the level of education of active workers in the field.

46. Applying these standards and for the purposes of this report, it is my opinion that one of skill in the art to which the Patents-in-Suit pertain in approximately mid-2009 would have had at least a bachelor's degree in Electrical Engineering, Computer Engineering, Applied Mathematics, or the equivalent, and at least two years of academic or industry experience in

computer architecture. The greater the educational background a person may have, the less work experience would be required, and vice versa. This person would have been capable of understanding and applying the teachings of the prior-art references discussed in this report.

V. TECHNICAL BACKGROUND

A. Floating-Point Numbers

47. Floating point refers to a way of representing numbers in which the location of the radix point—which is often colloquially referred to as the "decimal point" in base 10, and the "binary point" in binary notation, and which separates or delimits the fractional and integer parts of a number—may be different from number to number. The floating-point number format differs from fixed-point formats in which the location of the radix point is fixed within a given format (*e.g.*, for a fixed-point format that is only used to represent integer values, the radix point is assumed to be just to the right of the least significant digit). The term "floating point" derives from the fact that, in such formats, the radix point "floats," *i.e.*, is not in a predetermined position, within a particular representation. A floating-point number format typically consists of three fields or values that, collectively, are used to represent a real number *n*: (1) *sign*; (2) *exponent*; and (3) *fraction*. These three fields are used to represent *n* according to the following canonical formula (for binary floating-point formats):

$$(-1)^{sign} \times 1.fraction \times 2^{exponent-BIAS}$$

48. As the formula above shows, the *sign* field indicates the sign (positive or negative) of *n*. Together, the *exponent* and *fraction* fields determine the magnitude, or absolute value of *n*.

49. To better understand the floating-point number format, take the real number *n* = -62,239.5 as an example. The following formulae show how *n* would be encoded in a binary floating-point format with, for example, 1 bit for *sign*, 23 bits for *fraction*, and 8 bits for *exponent*, and an exponent bias of 127. The subscript 2 denotes values in base 2 (or binary):

$$-62{,}239.5 = (-1)^{1} \times 1.8993988037109375 \times 2^{15}$$

$$= (-1)^1 \times 1.8993988037109375 \times 2^{142-127}$$

$$= (-1_2)^1 \times 1.11100110001111110000000_2 \times 2^{10001110_2-127}$$

50. An advantage of floating-point number formats over fixed-point representations is in the range of possible values that can be represented for a given bitwidth. The use of the exponent in floating-point number formats plays an important role in enabling the larger range of possible values than would be possible using a fixed-point format with the same bitwidth.

51. The concept of a floating-point number format has been widely known and implemented since the earliest days in computing and has long been a fundamental concept taught as introductory material in most entry-level computer engineering or computer science courses, including well before the priority date of the Asserted Patents. For example, the *Computer Organization and Design* textbook by Patterson & Hennessy, which is a commonly used text in introductory computer engineering and science courses, has long provided a detailed overview of the floating-point number format.[13] The third edition of *Computer Organization and Design*, published in 2005, shows the canonical formula used to explicate floating-point number formats and explains the relationship between the fraction and exponent fields:[14]

In general, floating-point numbers are generally of the form

$$(-1)^S \times F \times 2^E$$

F involves the value in the fraction field and E involves the value in the exponent field; the exact relationship to these fields will be spelled out soon. (We will shortly see that MIPS does something slightly more sophisticated.)

These chosen sizes of exponent and fraction give MIPS computer arithmetic an extraordinary range. Fractions almost as small as $2.0_{ten} \times 10^{-38}$ and numbers almost as large as $2.0_{ten} \times 10^{38}$ can be represented in a computer. Alas, extraordinary differs from infinite, so it is still possible for numbers to be too large. Thus, overflow interrupts can occur in floating-point arithmetic as well as in integer arithmetic. Notice that **overflow** here means that the exponent is too large to be represented in the exponent field.

[13] *See* Patterson & Hennessy, COMPUTER ORGANIZATION AND DESIGN: THE HARDWARE/SOFTWARE INTERFACE § 3.6 (3d. ed. 2005) [GOOG-SING-00022145 at 22352-80].
[14] *See* Patterson & Hennessy, COMPUTER ORGANIZATION AND DESIGN: THE HARDWARE/SOFTWARE INTERFACE § 3.6 (3d. ed. 2005) [GOOG-SING-00022145 at 22355].

52. The *Computer Organization and Design* text also explains the historical pedigree of floating-point number formats:[15]

> These formats go beyond MIPS. They are part of the *IEEE 754 floating-point standard*, found in virtually every computer invented since 1980. This standard has greatly improved both the ease of porting floating-point programs and the quality of computer arithmetic.
>
> To pack even more bits into the significand, IEEE 754 makes the leading 1 bit of normalized binary numbers implicit. Hence, the number is actually 24 bits long in single precision (implied 1 and a 23-bit fraction), and 53 bits long in double precision (1 + 52). To be precise, we use the term *significand* to represent the 24- or 53-bit number that is 1 plus the fraction, and *fraction* when we mean the 23- or 52-bit number. Since 0 has no leading 1, it is given the reserved exponent value 0 so that the hardware won't attach a leading 1 to it.
>
> Thus $00\ldots00_{two}$ represents 0; the representation of the rest of the numbers uses the form from before with the hidden 1 added:
>
> $$(-1)^S \times (1 + \text{Fraction}) \times 2^E$$
>
> where the bits of the fraction represent a number between 0 and 1 and E specifies the value in the exponent field, to be given in detail shortly. If we number the bits

53. The earlier, second edition of *Computer Organization and Design*, published in 1998, provides similar discussions and explanations of floating-point number formats:[16]

> Computer arithmetic that supports such numbers is called *floating point* because it represents numbers in which the binary point is not fixed, as it is for integers. The programming language C uses the name *float* for such numbers. Just as in scientific notation, numbers are represented as a single nonzero digit to the left of the binary point. In binary, the form is
>
> $$1.xxxxxxxxx_{two} \times 2^{yyyy}$$
>
> (Although the computer represents the exponent in base 2 as well as the rest of the number, to simplify the notation we'll show the exponent in decimal.)

54. As noted in the various editions of *Computer Organization and Design*, the floating-point number format is ubiquitous enough that it is also the subject of an industry standard, namely IEEE 754, which was first adopted in 1985 and has been refined and re-affirmed periodically over the years since.[17]

---

[15] *See* Patterson & Hennessy, COMPUTER ORGANIZATION AND DESIGN: THE HARDWARE/SOFTWARE INTERFACE § 3.6 (3d. ed. 2005) [GOOG-SING-00022145 at 22356]

[16] *See* Patterson & Hennessy, COMPUTER ORGANIZATION AND DESIGN: THE HARDWARE/SOFTWARE INTERFACE § 4.8 (2d. ed. 1998) [GOOG-SING-00022064 at 22083-84].

[17] *See* IEEE 754-1985 - IEEE Standard for Binary Floating-Point Arithmetic, https://standards.ieee.org/ieee/754/993/ [LEESER000001].

1. Trade-Off Between Precision and Range

55. For a given floating-point number format (*i.e.*, for a given bitwidth of a floating-point format), there is a necessary trade-off between precision and range: the more bits in the format that are allocated to the exponent (thus increasing the range of possible values represented in the format), the fewer the bits that can be allocated to the fraction (thus decreasing the precision of possible values represented in the format), and vice versa.

56. As reflected in the above discussion regarding floating-point formats, "precision" refers to the number of significant figures with which a numeric type is stored; it is the part that is reflected by the fraction bits in the floating-point number format. The term "range" refers to the minimum and maximum value that numeric types can take; in the floating-point number format, it is reflected in the bits used for the exponent.

57. This inherent trade-off between precision and range was also widely known since the earliest days in computing and has long been a fundamental concept taught as introductory material in most entry-level computer engineering or science courses, including well before the priority date of the Asserted Patents. For example, the second edition of *Computer Organization and Design*, published in 1998, explains that the designer of a given floating-point representation "must find a compromise between the size of the significand and the size of the exponent because a fixed word size means you must take a bit from one to add a bit to the other," and that "[t]his trade-off is between accuracy and range: Increasing the size of the significand enhances the accuracy of the significand, while increasing the size of the exponent increases the range of numbers that can be represented":[18]

[18] Patterson & Hennessy, COMPUTER ORGANIZATION AND DESIGN: THE HARDWARE/SOFTWARE INTERFACE § 4.8 (2d. ed. 1998) [GOOG-SING-00022064 at 22084]; *see also* Patterson & Hennessy, COMPUTER ORGANIZATION AND DESIGN: THE HARDWARE/SOFTWARE INTERFACE § 3.6 (3d. ed. 2005) [GOOG-SING-00022145 at 22354].

Floating-Point Representation

The designer of a floating-point representation must find a compromise between the size of the significand and the size of the exponent because a fixed word size means you must take a bit from one to add a bit to the other. This trade-off is between accuracy and range: Increasing the size of the significand enhances the accuracy of the significand, while increasing the size of the exponent increases the range of numbers that can be represented. As our design guideline from Chapter 3 reminds us, good design demands good compromises.

2. Mantissa

58. The mantissa of a number represented in a floating-point number format, sometimes referred to as the significand, is the portion that represents the significant digits of that number, and that is multiplied by the base (2 in binary) raised to the exponent to give the actual value of the number. In the example above, describing a floating-point representation for the number -62,239.58, the mantissa (in binary) is 1.11100110001111110010100. In this example, the mantissa has only one non-zero integer digit, which is referred to as normalized form. As discussed in more detail below, floating-point numbers are generally normalized, meaning that the mantissa (again, in binary) is of the form 1.xxxx. Since the only non-zero digit in binary is '1', the leading bit in the mantissa of a normalized floating-point number is always '1', and that bit becomes unnecessary to store. Instead, only the fractional part—the xxxx part of 1.xxxx—is stored.

3. Exponent

59. The exponent field represents the power that 2 is raised to in the floating point number. For a floating-point number format that follows the IEEE biasing scheme with *n* bits used to represent the exponent, the range of exponents that can be represented are $-2^{(n-1)} + 1$ to $2^{(n-1)}$. In addition, IEEE floating-point number formats reserve the smallest and largest exponents (all zeros or all ones) to represent special values. For example, in the IEEE single-precision floating-point format, which has 1 sign bit, 8 exponent bits, and 23 mantissa bits, the range of exponents that can be represented are -126 to 127.

4. Bias

60. As noted above, a concept closely related to the exponent field of a floating-point number is that of bias. In general, when using a biasing scheme similar to that used in the IEEE single-precision format, the value of the bias depends on the exponent bitwidth (*i.e.*, the number of bits used to represent the exponent) according to the following formula, where *n* is the exponent bitwidth:

$$bias = 2^{n-1} - 1$$

61. Thus, in this kind of biasing scheme, the bias will be 127 when the exponent is 8 bits wide, the bias will be 63 when the exponent is 7 bits, the bias will be 31 when the exponent is 6 bits, and so on.

5. Dynamic Range

62. As can be seen, a larger bitwidth (*i.e.*, number of bits) for the exponent in a floating-point representation allows both larger numbers and smaller numbers to be represented as compared to a smaller exponent bitwidth. For example, even assuming that the smallest and largest exponents (all zeros or all ones) are reserved to represent special values like in IEEE convention, a 5-bit exponent allows representing real numbers as large (in absolute value) as *mantissa* x 2^{15} (where $2^{15} = 32{,}768$) and numbers as small as *mantissa* x 2^{-14} (where $2^{-14} = 1 / 16{,}384$). The span of possible numbers represented in a particular number format is referred to as the dynamic range of the number format (or sometimes just "range"). Adding one more bit to the exponent bitwidth (*i.e.*, for a 6-bit biased exponent) allows the exponent to range from -30 to 31 and thus allows representing numbers as large as *mantissa* x 2^{31} (where $2^{31} = 2{,}147{,}483{,}648$) and numbers as small as *mantissa* x 2^{-30} (where $2^{-30} = 1 / 1{,}073{,}741{,}824$). Thus, as can be seen, an increase of the bitwidth of the exponent in a floating point number format by just one additional bit can dramatically increase the dynamic range of the format.

63. The fact that, for a given bitwidth, the floating-point number format enables much greater dynamic range than fixed-point number format has been well known for many years before the priority date of the Asserted Patents. It was also well-recognized long before the

priority date of the Asserted Patents that, for a given bitwidth floating-point format, there is a tradeoff between precision and range due to the fact that more bits allocated to the mantissa means fewer bits available for the exponent (leading to greater precision at the expense of dynamic range), and, conversely, more bits allocated to the exponent means fewer bits available for the mantissa (leading to greater dynamic range at the expense of precision). As just one example, the same 1998 edition of the Patterson & Hennessy text explains: "The designer of a floating-point representation must find a compromise between the size of the significand [mantissa] and the size of the exponent, because a fixed word size means you must take a bit from one to add a bit to the other. This trade-off is between accuracy and range: Increasing the size of the significand [mantissa] enhances the accuracy of the significand [mantissa], while increasing the size of the exponent increases the range of numbers that can be represented."[19]

6. Normalization

64. Another important facet of floating-point numbers is the concept of normalization. Normalizing refers to the process of standardizing a number represented in floating-point so that the mantissa has only one non-zero digit to the left of the radix point. For example, using base-10 representation for simplicity, $3.14159265359 \times 10^8$ would be the normalized form of the real number 314,159,265.359; whereas $31.4159265359 \times 10^7$ would not be the normalized form, because the latter representation contains two non-zero digits to the left of the radix point rather than just one. In binary, the only non-zero digit is '1', so a normalized floating-point value represented in base-2 will have exactly one '1' to the left of the radix point. Because this will always be the case for a normalized floating-point value, the '1' becomes redundant information, is unnecessary to store, and is therefore usually not stored. Instead, in what has long been a common technique, only the fractional digits of the mantissa are stored, which is why the term "fraction" is often used to refer to the field containing the mantissa bits of a floating-point number. However, the leading '1' is still understood to be part of the mantissa

[19] Patterson & Hennessy, COMPUTER ORGANIZATION AND DESIGN: THE HARDWARE/SOFTWARE INTERFACE § 4.8 (2d. ed. 1998) [GOOG-SING-00022064 at 22084].

and is therefore commonly referred to as an "implied" '1' or "hidden bit." That implied '1' must be added back or restored to the representation of the floating-point value in order to perform further processing.

B. Field-Programmable Gate Arrays

65. A Field-Programmable Gate Array (FPGA) is an integrated circuit that is designed to be configurable by a designer *after* manufacturing, which is in contrast to general purpose computing architectures (*e.g.*, microprocessors) that use hardware elements with a fixed architecture. This flexibility gives FPGAs certain traits of software, in that the designer can customize the structure of the FPGA to accelerate application-specific tasks, while enabling the speed of specialized hardware.

66. Code specifying the desired structure of an FPGA is typically written in a Hardware Description Language (HDL) such as VHDL (VHSIC Hardware Description Language) or Verilog. To perform mapping of the HDL code to the logic-gate level of an FPGA, the hardware description is synthesized, mapped, placed, and routed. The synthesis process generates an intermediate format that is mapped to individual Configurable Logic Blocks (CLBs) on the FPGA, and the CLBs are placed at specific locations on the chip and routed. The result is a bitstream that can be downloaded and run on the FPGA.

67. CLBs are the fundamental components of an FPGA and contain the functional elements for constructing logic. The CLBs generally allow the designer to implement virtually any logical functionality within the chip. Each CLB generally contains multiple "slices" that contain various logic elements (*e.g.*, flip-flops, look-up tables, multiplexers, etc.).

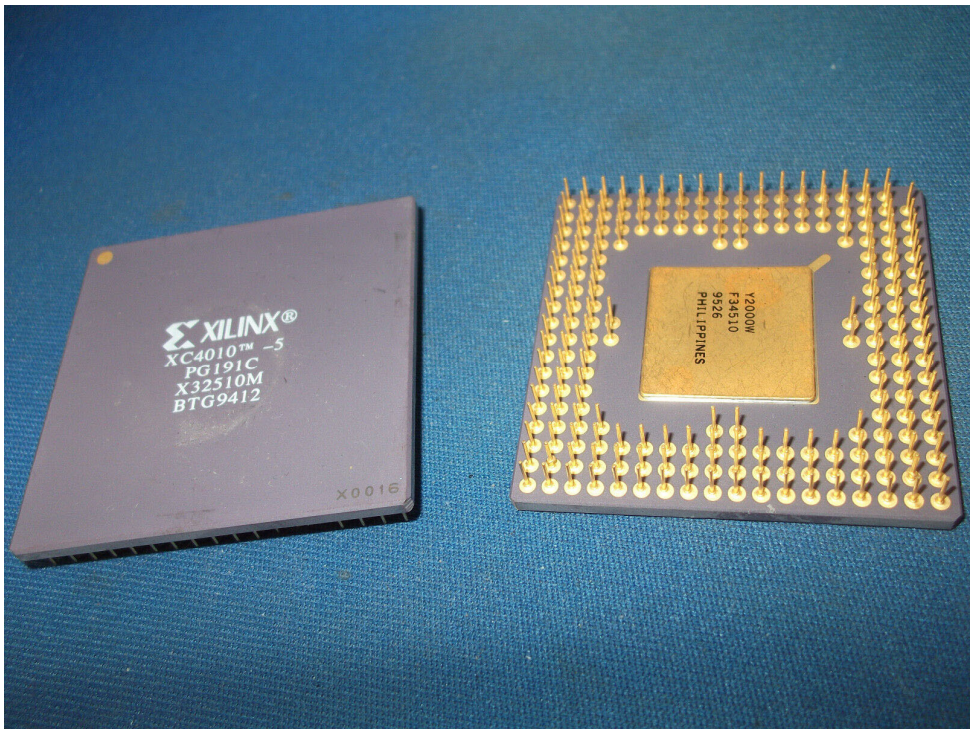
68. The final step of generating the bitstream that can be downloaded and run on the FPGA is usually done by tools from the FPGA manufacturer, in our case Xilinx. A number of commercially available tools exist to perform synthesis, mapping, and sometimes placement, and routing of HDL description onto an FPGA, including tools from Synplicity that were used to

generate designs for the VFLOAT library discussed below. Once the bitstream has been generated it is downloaded to the FPGA.

69. FPGAs have been around since the mid-1980s, when Xilinx, Inc., introduced the first one, the Xilinx XC2064 (seen below), in 1984.[20]

70. As one would expect, FPGA technology has progressed steadily and considerably since that time. For example, the XC2064 had just 64 CLBs, the equivalent of less than 1,000 gates—800 to be exact—and used a 2µ (micron) process technology.[21] Xilinx's flagship FPGA in the early 1990s, the XC4010 (seen below), employed 400 CLBs for a maximum of 10,000 gate equivalents.[22]

[20] *See* Steve Trimberger, *Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology* at 321, Proceedings of the IEEE, vol. 103, no. 3, pp. 318-331, March 2015, *available at* https://ieeexplore.ieee.org/document/7086413 [LEESER000027 at 30].
[21] *See* Steve Trimberger, *Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology* at 321, Proceedings of the IEEE, vol. 103, no. 3, pp. 318-331, March 2015, *available at* https://ieeexplore.ieee.org/document/7086413 [LEESER000027 at 30]; Xcell, Vol. 32 at 4 (Q2 1999), available at https://www.xilinx.com/publications/archives/xcell/Xcell32.pdf [LEESER000041 at 44].
[22] *See* Steve Trimberger, *Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology* at 323, Proceedings of the IEEE, vol. 103, no. 3, pp. 318-331, March 2015, *available at* https://ieeexplore.ieee.org/document/7086413 [LEESER000027 at 32]; Xilinx XC4010D, XC4013D Datasheet at 1, *available at* https://www.semiee.com/file/EOL/Xilinx-XC4000D.pdf [LEESER000105].



71. By 1999, when Xilinx introduced its Virtex XCV1000 FPGA, the device had 6,144 CLBs or the equivalent of over 1,000,000 (1,124,022) system gates.[23] By 2006, when Xilinx introduced its Virtex-5 family of FPGAs, an FPGA might have in excess of 25,000 CLBs and 9,000,000 gate equivalents. For example, Xilinx's Virtex-5 family XC5VLX330T FPGA had 25,920 CLBs and approximately 9,483,935 gate equivalents.[24]

72. Devices that incorporated multiple FPGAs also steadily grew in sophistication as well. For example, the first-generation "WILDSTAR" reconfigurable computing engine from Annapolis Micro Systems, Inc. ("Annapolis Micro"), which was first announced in 1999, featured 3 Xilinx VIRTEX XCV1000 FPGAs. By 2008, Annapolis Micro's latest-generation

[23] *See* Steve Trimberger, *Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology* at 323, Proceedings of the IEEE, vol. 103, no. 3, pp. 318-331, March 2015, *available at* https://ieeexplore.ieee.org/document/7086413 [LEESER000027 at 32]; Virtex™ 2.5 V Field Programmable Gate Arrays Datasheet at 1 (v1.7, Oct. 1, 1999) [XILINX-GOOG-SUB00000069 at 69]; Virtex™ 2.5 V Field Programmable Gate Arrays Datasheet at 1 (v2.5, April 2, 2001) [XILINX-GOOG-SUB00000522 at 522].

[24] *See* Virtex-5 Family Overview at 2 (v4.4, Sept. 23, 2008), *available at* https://inst.eecs.berkeley.edu/~cs150/sp12/resources/ds100.pdf [LEESER000107 at 108].

reconfigurable computing engine, WILDSTAR 5, featured 3 Xilinx Virtex-5 FPGAs, such as three Xilinx Virtex-5 XC5VLX330T FPGAs.[25]

73. In addition to the physical WILDSTAR board itself, Annapolis Micro provided software referred to as a board support package (BSP) that ran on the x86 host machine. The BSP supported downloading the bitstream from the host, runtime communication with the host, and communication with on-board memory.

VI. THE VFLOAT LIBRARY

74. In this section, I provide an overview of the VFLOAT library, including what VFLOAT was, the origins or motivation that originally led to the development of VFLOAT, who was involved in the work, our deployment of VFLOAT on reconfigurable computing hardware, and the information that I and others made publicly available about VFLOAT and related work.

A. VFLOAT Overview / Background

75. VFLOAT, also known more descriptively as "The Northeastern Variable-Precision Floating Point Library," was a library (or set) of parameterized hardware modules for performing arithmetic operations on floating-point numbers of various formats using reconfigurable hardware, namely FPGAs. What that means is that a designer using VFLOAT could quickly deploy, onto FPGA hardware, circuits containing multiplication units and other floating-point arithmetic modules that operate on a variety of customizable floating-point number formats. The designer simply selects the floating-point number format they wish to use by specifying bitwidths for the exponent and fraction; the code in the library could then create the corresponding VFLOAT arithmetic modules for implementation on FPGAs.

76. Our general motivation in developing VFLOAT was the idea or understanding that, when implementing floating-point arithmetic operators on reconfigurable hardware such as FPGAs, fine-grained control over the floating-point formats used is possible and even desirable because the optimal bitwidth of a signal often depends on the particular application, and the

[25] *See* https://web.archive.org/web/20081004190813/http://www.annapmicro.com/ws5pci.html [LEESER000120].

further idea or understanding that flexible control over bitwidths allows greater parallelism and more efficient use of hardware resources.

77. VFLOAT was developed by myself and others in the 2000-2002 timeframe at the Reconfigurable Computing Laboratory (RCL) at Northeastern University (back when it was known as the Rapid Prototyping Laboratory or RPL), the group that I have led since 1996, including during the 2000-20002 timeframe when VFLOAT was first developed. Specifically, in addition to myself, other contributors to the original VFLOAT library included Pavle Belanović, who at the time was a graduate student in Electrical & Computer Engineering at Northeastern University whose graduate work I supervised. Several other graduate students whom I supervised also contributed to subsequent iterations or revisions of VFLOAT over the years, including Haiqian Yu in 2003 and Xiaojun Wang in 2008, although the focus of this report is the original version of VFLOAT that was developed and made publicly available in approximately 2002.

78. The origins or genesis of VFLOAT can be traced to my work for Los Alamos National Laboratory (LANL), in approximately 1999-2002. LANL, which is located just outside Santa Fe, New Mexico, is one of a few national laboratories operated by the U.S. Department of Energy and is most well known for first being organized during World War II for the design of nuclear weapons as part of the Manhattan Project. I began working for LANL in approximately 1999 on a sub-contract received by Northeastern University, which was funded by a grant from the U.S. Defense Advanced Research Projects Agency (DARPA). DARPA is the agency of the U.S. Department of Defense responsible for researching and developing emerging technologies for use by the military, though those technologies often end up having significant application in the commercial or private sector. The Internet is probably the most well-known example of a technology originally developed by DARPA; its precursor was DARPA's ARPANET, the first wide-area packet-switched computer network.

79. The formal title of the sub-contract with LANL was "Acceleration of Scene Classification and Spectral Unmixing with Reconfigurable Computing," and is identified on my

CV under "Research Grants and Contracts." In general, my objective for this sub-contract with LANL was to explore ways to use FPGAs to quickly analyze large volumes of satellite data that LANL received and needed to process, and more specifically to accelerate their ability to analyze this data compared to existing techniques. This project for LANL supported both myself and several of my students at the time, including Mr. Belanović, who (as mentioned earlier) was a graduate student pursuing his Masters of Science degree in Electrical Engineering at Northeastern University. The individuals we worked and interfaced with at LANL were officials in the Space and Remote Sensing Sciences Group.

80. VFLOAT became the deliverable for our work for LANL. Under my guidance, Mr. Belanović developed a library of hardware modules for performing variable-precision floating-point arithmetic on FPGAs, which we later named VFLOAT—the "FLOAT" signifying that the modules in the library were designed to perform arithmetic operations using floating-point numbers and the "V" signifying that the modules were *variable* in that they used custom, user-defined floating-point number formats (that is, floating-point formats that could have different numbers of exponent and fraction bits as compared to the IEEE standard single-precision format).

81. The initial version of VFLOAT was developed specifically for our work for LANL, and LANL directed us to open-source (*i.e.*, non-classified) data sets to validate the use of VFLOAT for their specific application (analyzing satellite imagery and other data). That is also the reason why a number of our presentations and papers related to VFLOAT over the years identify satellite data processing as an example application of VFLOAT. For example, in my presentation at HPEC 2002 (discussed in more detail below), I presented on, among other things, a hybrid implementation of the K-means clustering algorithm using VFLOAT in the specific context of satellite imagery. This is reflected in the slides I used for that HPEC 2002 presentation. *See* **Exhibit C**.

82. At an earlier stage in our work for LANL, we had developed an implementation of K-means clustering[26] in reconfigurable hardware (FPGAs) called "ASAPP" that proved to be orders of magnitude faster than implementing the same algorithm in software. But because it was designed for one dataset with a particular set of parameters (multispectral images with 10 channels, 12 bits per channel, and 8 clusters, to match MTI data[27]), we subsequently developed a parameterized hardware implementation of K-means clustering to more easily adapt to various different datasets available within the domain of multispectral or hyperspectral satellite imagery (or other domains)—for example, datasets with a different number of channels, with a different number of bits per channel, with a different number of pixels, and/or requiring a different number of clusters.[28] VFLOAT was very much a logical extension of this work in that it carried the concept of parameterization into the underlying floating-point arithmetic modules used to implement K-means clustering.

83. As discussed in more detail below, I personally traveled to New Mexico several times during the course of our work with the Space and Remote Sensing Sciences Group to discuss and present our work related to VFLOAT to the officials there, and Mr. Belanović accompanied me on at least one occasion. Based on my interactions with the Space and Remote Sensing Sciences Group, I believe that LANL ultimately deployed VFLOAT for the purpose of analyzing satellite imagery and other data using actual FPGA hardware.

---

[26] K-means clustering is a common technique for segmentation of multi-dimensional data, such as multispectral and hyperspectral satellite imagery.

[27] MTI refers to the "Multispectral Thermal Imager," which was a Department of Energy-funded satellite in polar orbit that was launched in early 2000 and carried an advanced multispectral and thermal imaging sensor.

[28] This work on a parameterized hardware implementation of K-means clustering is described in more detail in "Applying Reconfigurable Hardware to the Analysis of Multispectral and Hyperspectral Imagery" , a paper that I co-authored along with Mr. Belanović, Michael Estlick, and three individuals from LANL with whom we worked (Maya Gokhale, John Szymanski, and James Theiler), and which I presented at the 2001 International Symposium on Optical Science and Technology. *See Proceedings - SPIE*, Vol. 4480 (2001), https://www.spiedigitallibrary.org/conference-proceedings-of-spie/4480/0000/Applying-reconfigurable-hardware-to-the-analysis-of-multispectral-and-hyperspectral/10.1117/12.453329.short [LEESER000122].

B. Detailed Discussion of VFLOAT

84. As described above, FPGAs are a form of reconfigurable hardware whose structure or architecture can be specified by a designer after manufacturing, typically using a Hardware Description Language (HDL). In the case of VFLOAT, we programmed the design of the various hardware modules in VHDL (VHSIC Hardware Description Language). When mapped to FPGA hardware, VFLOAT modules define circuits that are structured and behave as specified by the VHDL code. The full set of code comprising the original VFLOAT library is attached to my report as **Exhibit D**.

1. Hardware Modules Comprising VFLOAT

85. At a high level, VFLOAT comprises parameterized floating-point arithmetic modules, defined in VHDL, for performing floating-point addition (*fp_add*[29]) and floating-point subtraction (*fp_sub*[30]), as well as a module that performs the exponent addition and mantissa multiplication steps used in floating-point multiplication (*fp_mul*[31]). VFLOAT also comprises a number of low-level hardware modules that are used as building blocks in these top-level arithmetic operators, such as a fixed-point multiplier (*parameterized_multiplier*[32]), which is needed, for example, to perform mantissa multiplication. VFLOAT also comprises a number of format control modules, such as a denormalization module (*denorm*[33]), which adds back the implied '1' to the fraction field to form a mantissa with a non-zero integer part, and a rounding and normalization module (*rnd_norm*[34]), which performs rounding and normalization of floating-point values created by the arithmetic modules. VFLOAT further comprises various other categories of modules, such as parameterized modules for converting from fixed-point to floating-point number formats (*fix2float*[35]) and vice versa (*float2fix*[36]).

---

[29] *See* Exhibit D at fp_add.vhd.
[30] *See* Exhibit D at fp_sub.vhd.
[31] *See* Exhibit D at fp_mul.vhd.
[32] *See* Exhibit D at parameterized_multiplier.vhd.
[33] *See* Exhibit D at denorm.vhd.
[34] *See* Exhibit D at rnd_norm.vhd.
[35] *See* Exhibit D at fix2float.vhd.
[36] *See* Exhibit D at float2fix.vhd.

86. As detailed below, the arithmetic modules comprising VFLOAT accept exponents that are biased using a biasing scheme similar to that used in the IEEE single-precision format.

2. Parameterization of VFLOAT Hardware Modules

87. As noted above, the floating-point arithmetic modules comprising VFLOAT are parameterized, which means that they could operate on essentially arbitrary floating-point number formats selected by the designer. In fact, nearly all the modules (including the low-level and format-control modules discussed above) that comprise the VFLOAT library are parameterized. This parameterization can be seen directly in the source code for VFLOAT.

88. For example, the module *fp_mul* (which, as noted above, is the module that performs the exponent addition and mantissa multiplication steps used in floating-point multiplication) has two floating-point number inputs, `OP1` and `OP2`.[37] The VHDL `port` declaration[38] for *fp_mul* explicitly defines those inputs (`OP1` and `OP2`) based on two parameters: the bitwidth of the exponent for the selected floating-point format (`exp_bits`) and the bitwidth of the mantissa for the selected floating-point format (`man_bits`). Specifically, the inputs `OP1` and `OP2` are each defined as *n*-element vectors, using the `std_logic_vector` data type, where n = `exp_bits` + `man_bits` + 1).[39] These parameters (`exp_bits` and `man_bits`), in turn, are defined in the `generic` clause of *fp_mul*, where they can be customized in accordance with the designer's specific application by specifying the exponent bitwidth and mantissa bitwidth.[40] When instantiated, the *fp_mul* module creates a circuit on the FPGA that accepts floating-point inputs with the exponent and mantissa bitwidths specified by `exp_bits` and `man_bits`.

---

[37] *See* Exhibit D at fp_mul.vhd:65-77.

[38] "In a VHDL Output File (. vho), a port name in the Entity Declaration represents an input or output of the current file. When an instance of a primitive or lower-level design file is implemented with a Component Instantiation, its ports are connected to signals with Port Map Aspects." *See* https://www.intel.com/content/www/us/en/programmable/quartushelp/13.0/mergedProjects/reference/glossary/def_port.htm#:~:text=In%20a%20VHDL%20Output%20File,signals%20with%20Port%20Map%20Aspects [LEESER000157].

[39] *See* Exhibit D at fp_mul.vhd:68-69.

[40] *See* Exhibit D at fp_mul.vhd:60-64.

89. Similarly, the module *denorm* (which, as noted above, is the module that performs denormalization of floating-point values) has a floating-point input, `IN1`, on which it performs the denormalization operation, and a floating-point output, `OUT1`, to output the results of the denormalization.[41] The VHDL `port` declaration for *denorm* explicitly defines the input (`IN1`) and output (`OUT1`) based on two parameters: the bitwidth of the exponent for the selected floating-point format (`exp_bits`) and the bitwidth of the mantissa for the selected floating-point format (`man_bits`). Specifically, input `IN1` is defined as an *n*-element vector using the `std_logic_vector` data type, where *n* = `exp_bits` + `man_bits` + 1 (the additional bit beyond the sum of parameters `exp_bits` and `man_bits` is for the sign bit[42]) and output `OUT1` is defined as an *n*+1-element vector (to accommodate the implied '1' added to create the mantissa).[43] These parameters (`exp_bits` and `man_bits`), in turn, are defined in the `generic` clause of *denorm*, where they can be customized in accordance with the designer's specific application by specifying the exponent bitwidth and mantissa bitwidth.[44] When instantiated, the *denorm* module creates a circuit on the FPGA that accepts a floating-point input with the exponent and mantissa bitwidths specified by `exp_bits` and `man_bits` and adds back the implied '1' to form a mantissa with a non-zero integer part.

90. Likewise, the module *rnd_norm* (which, as noted above, is the module that performs rounding and normalization of floating-point values) has a floating-point input, `IN1`, on which it performs normalization and rounding, and a floating-point output, `OUT1`, to output the results of the normalization and rounding.[45] The VHDL `port` declaration for *rnd_norm* explicitly defines the input (`IN1`) based on two parameters: the bitwidth of the exponent for the selected floating-point format (`exp_bits`) and the bitwidth of the input mantissa (`man_bits_in`). Specifically, the input `IN1` is defined as an *n*-element vector using the

[41] *See* Exhibit D at denorm.vhd:60-70.
[42] *See* Exhibit D at denorm.vhd:87 (connecting sign signal to most-significant bit of `IN1`: `s_old <= IN1(man_bits+exp_bits))`.
[43] *See* Exhibit D at denorm.vhd:63, 67.
[44] *See* Exhibit D at denorm.vhd:55-59.
[45] *See* Exhibit D at rnd_norm.vhd:61-73.

`std_logic_vector` data type, where n = `exp_bits` + `man_bits_in` + 1.[46] Likewise, the VHDL `port` declaration for *rnd_norm* explicitly defines the output (`OUT1`) based on two parameters: the bitwidth of the exponent for the selected floating-point format (`exp_bits`) and the bitwidth of the output mantissa (`man_bits_out`). Specifically, the output `OUT1` is defined as an m-element vector using the `std_logic_vector` data type, where m = `exp_bits` + `man_bits_out` + 1.[47] These parameters (`exp_bits`, `man_bits_in`, and `man_bits_out`), in turn, are defined in the `generic` clause of *rnd_norm*, where they can be customized in accordance with the designer's specific application by specifying the exponent bitwidth and mantissa bitwidths for the input and output.[48] When instantiated, the *rnd_norm* module creates a circuit on the FPGA that accepts a floating-point input with the exponent and mantissa bitwidths specified by `exp_bits` and `man_bits_in` and outputs a rounded and normalized floating-point value (*i.e.*, the floating-point input after rounding and normalization) with the exponent and mantissa bitwidths specified by `exp_bits` and `man_bits_out`.

3. VFLOAT Multiplier

91. As noted, the VFLOAT library was comprised of parameterizable modules that were designed to be—and were—used in combination, kind of like Lego building blocks for programming FPGA circuits. For example, 2 *denorm*, 1 *fp_mul*, and 1 *rnd_norm* would be assembled to create a complete multiplier circuit.[49]

92. The size of the various floating-point hardware modules created on FPGAs using the VFLOAT library, including *denorm*, *fp_mul*, and *rnd_norm*, was dependent on the bitwidth of the floating-point numbers those modules operated on. In the multiplier hardware created on FPGAs by the VFLOAT library, the overall size of a floating-point multiplier is dominated by the hardware needed to multiply the mantissas of two numbers; thus, for a given total bitwidth,

[46] *See* Exhibit D at rnd_norm.vhd:64.
[47] *See* Exhibit D at rnd_norm.vhd:70.
[48] *See* Exhibit D at rnd_norm.vhd:55-60
[49] *See* Exhibit C at 11 (HPEC 2002 Slides) (describing assembly of modules for an IEEE single-precision adder).

allocating fewer bits to the fraction field and more to the exponent field reduces the overall size of the multiplier.

a. VFLOAT Floating-Point Denormalization Module: *denorm*

93. As noted above, the parameterized module in VFLOAT for performing denormalization on floating-point values (*i.e.*, adding the implied '1' to the fraction field to form a mantissa with a non-zero integer part) was named *denorm*, and its design (*i.e.*, structure and behavior) is expressed in the VHDL source code specified in denorm.vhd.[50] The *denorm* module accepts as input one floating-point number and outputs the denormalized form of that number. As can be seen in the source code for *denorm* and in particular its VHDL `port` declaration, the input is denoted `IN1` and the output is denoted `OUT1`.[51]

94. The design of the *denorm* module is relatively straightforward, as reflected in the VHDL code. The *denorm* module copies or forwards the input's (`IN1`) sign bit to the sign bit of the output (`OUT1`).[52] Similarly, the *denorm* module copies or forwards the exponent bits of the input (`IN1`) to the exponent bits of the output (`OUT1`).[53] For the mantissa, the *denorm* module copies or forwards the fraction bits of the input (`IN1`) to the corresponding mantissa bits of the output (`OUT1`) and adds a leading '1' (*i.e.*, the implied '1') to the most-significant bit of the output mantissa.[54] To accommodate the insertion of the additional bit in the output mantissa, the output of the *denorm* module (`OUT1`) is one bit wider than than the input (`IN1`), and the *denorm* module internally uses fraction signals that are one bit wider for the output mantissa (`f_new`) than the input fraction (`f_old`).[55] When the input floating-point number is '0', which is represented according to IEEE convention by all zero exponent and fraction/mantissa bits, the

[50] *See* Exhibit D at denorm.vhd.
[51] *See* Exhibit D at denorm.vhd:60-70.
[52] *See* Exhibit D at denorm.vhd:77-78, 87, 104, 110.
[53] *See* Exhibit D at denorm.vhd:79-80, 88, 105, 111.
[54] *See* Exhibit D at denorm.vhd:81-82, 89, 92-101, 106-107, 112.
[55] *See* Exhibit D at denorm.vhd:63, 67 (defining `IN1` as a vector with `exp_bits` + `man_bits` + 1 elements, and OUT1 as a vector with `exp_bits` + `man_bits` + 2 elements); Exhibit D at denorm.vhd:81-82 (defining `f_old` as a vector with `man_bits` elements, and `f_new` as a vector with `man_bits` + 1 elements).

denorm module inserts a '0' bit rather than the implied '1' at the most-significant bit of the output mantissa, so that the output remains '0'. To accommodate this scenario, the *denorm* module checks the input exponent bits using a simple *n*-input OR gate that is built from the lower-level hardware module *parameterized_or_gate*.[56] If any of those bits are '1', the OR gate will output a '1', which is used to insert the implied '1' into the most-significant bit of the output mantissa; otherwise the output of '0' from the OR gate will be used to insert a '0' into the most-significant bit of the output mantissa.[57]

b. VFLOAT Exponent Addition and Mantissa Multiplication Module: *fp_mul*

95. As noted above, the parameterized arithmetic module in VFLOAT for performing exponent addition and mantissa multiplication was named *fp_mul*, and its design (*i.e.*, structure and behavior) is expressed in the VHDL source code specified in fp_mul.vhd.[58] The module *fp_mul* accepts as inputs two floating-point numbers and outputs the sum of their exponents and the product of their mantissas. As can be seen in the source code for *fp_mul* and in particular its VHDL `port` declaration (which defines the interfaces for a VHDL entity, *i.e.*, the inputs and outputs to/from a block), those inputs are denoted `OP1` and `OP2`.[59] Similarly, the output is denoted in the `port` declaration as `RESULT`.[60]

96. The module *fp_mul* includes a fixed-point adder (denoted `exponent_adder` in the code), which is built from the lower-level hardware module *parameterized_adder*[61] and sums together the exponents of the two floating-point inputs.[62] The *fp_mul* module also includes a fixed-point subtractor (denoted `bias_subtractor` in the code), which is built from the lower-level hardware module *parameterized_subtractor*[63] and subtracts the exponent bias from

[56] *See* Exhibit D at denorm.vhd:92-101; Exhibit D at parameterized_or_gate.vhd.
[57] *See* Exhibit D at denorm.vhd:88, 92-101, 107, 112.
[58] *See* Exhibit D at fp_mul.vhd.
[59] *See* Exhibit D at fp_mul.vhd:65-69.
[60] *See* Exhibit D at fp_mul.vhd:65-74.
[61] *See* Exhibit D at parameterized_adder.vhd.
[62] *See* Exhibit D at fp_mul.vhd:127-139, 198, 201.
[63] *See* Exhibit D at parameterized_subtractor.

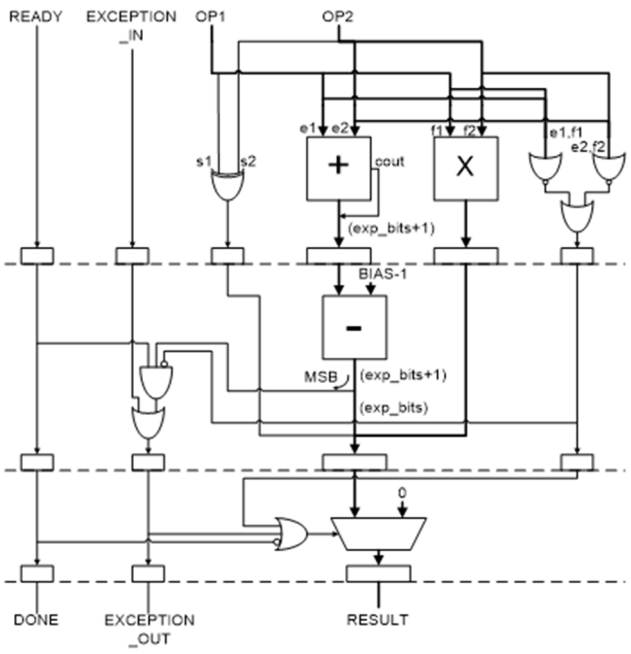
the sum of the input exponents in order to account for the fact that the input exponents are biased.[64] The module *fp_mul* also includes a fixed-point multiplier (denoted `mantissa_multiplier` in the code), which is built from the lower-level hardware module *parameterized_multiplier*[65] and multiplies the mantissas of the two floating-point inputs.[66]

97. The *fp_mul* module also includes a number of registers used for timing purposes.[67] As can be seen in the source code for *fp_mul* and in particular its VHDL `port` declaration, the module also accepts a control signal, denoted `READY` in the code, that instructs it to begin processing data.[68]

98. The *fp_mul* module also receives an exception signal (`EXCEPTION_IN`) and can output its own exception signal (`EXCEPTION_OUT`), as can also be seen in the source code.[69] One circumstance in which the value of `EXCEPTION_OUT` will be set to 1, to signal an exception, is if the output of the *paramterized_subtractor* was either too large or too small to be represented in the exponent field of the output.

99. When implemented in hardware, an *fp_mul* module creates a circuit on the FPGA with the following structure:[70]

[64] *See* Exhibit D at fp_mul.vhd:151-161, 224.
[65] *See* Exhibit D at parameterized_multiplier.vhd.
[66] *See* Exhibit D at fp_mul.vhd:140-150, 199, 202.
[67] *See* Exhibit D at fp_mul.vhd:215-235 (specifying synchronous assignment of certain timing registers).
[68] *See* Exhibit D at fp_mul.vhd:65-77.
[69] *See* Exhibit D at fp_mul.vhd:65-77.
[70] This diagram is taken from the slides for my presentation at HPEC 2002 (which is discussed in more detail below).



100. The `RESULT` output of the *fp_mul* module is designed to accommodate the same number of exponent bits as the inputs but twice the number of mantissa bits as the inputs, because of the mathematical fact that performing the fixed-point multiplication of the mantissas results, at least initially (*i.e.*, before any rounding), in a product mantissa that is wider than the mantissa of the inputs (as noted above, the output mantissa is calculated as the fixed-point products of the input mantissas).[71] Returning the output of *fp_mul* to the same bitwidth as the inputs to the multiplication operation is one function of the *rnd_norm* module, discussed in more detail below.

101. As discussed above, VFLOAT arithmetic modules were designed to accept input exponents that were biased using a biasing scheme similar to that used in the IEEE single-precision format. As discussed above, in such a biasing scheme, a bias is applied to determine the stored exponent value, and the value of that bias that depends on the bitwidth of the exponent. As discussed above, assuming an *n*-bit biased exponent, the bias can be formulaically expressed as: $2^{n-1} - 1$. For example, in the case of a 6-bit biased exponent, the bias is $2^{6-1} - 1 =$

[71] *See* Exhibit D at fp_mul.vhd:74 (defining `RESULT` as `std_logic_vector(exp_bits+(2*man_bits) downto 0)`).

2^5 - 1 = 32 - 1 = 31. In the case of the *fp_mul* module, the use of a biasing scheme can be seen, for example, in the use of the `bias_subtractor`, which takes the sum of the input exponents and subtracts one copy of the bias from that sum.[72]

c. VFLOAT Floating-Point Rounding and Normalization Module: *rnd_norm*

102. As noted above, the parameterized module in VFLOAT for performing normalization and rounding on floating-point values was named *rnd_norm*, and its design (*i.e.*, structure and behavior) is expressed in the VHDL source code specified in rnd_norm.vhd.[73] The *rnd_norm* module accepts as input one floating-point number, performs normalization on that number, rounds it, then outputs the result.

103. As can be seen in the source code for *rnd_norm* and in particular its VHDL `port` declaration, the input is denoted `IN1` and the output is denoted `OUT1`.[74] As discussed above, the input `IN1` and the output `OUT1` are parameterized based on the values specified by `exp_bits`, `man_bits_in`, and `man_bits_out`. Specifically, the parameter `man_bits_in` specifies the mantissa bitwidth for the input `IN1`, the parameter `man_bits_out` specifies the mantissa bitwidth for the output `OUT1`, and the parameter `exp_bits` specifies the exponent bitwidth for both the input `IN1` and the output `OUT1`.[75] The reason that different parameters are used to specify the mantissa bitwidths for the input and output of *rnd_norm* is that the rounding operation performed by *rnd_norm* entails a reduction in the mantissa bitwidth. The difference between `man_bits_in` and `man_bits_out` corresponds to the number of bits to be eliminated from the mantissa in the rounding and normalization process. As can also be seen in the source code for *rnd_norm* and in particular its VHDL `port` declaration, the module also accepts a control signal, denoted `READY` in the code, that instructs it to begin processing data.[76]

---

[72] *See* Exhibit D at fp_mul.vhd:137, 151-161, 224. The exponent bias is computed previously in the code for *fp_mul* based on the bitwidth of the input exponents. *See* Exhibit D at fp_mul.vhd:212-214.
[73] *See* Exhibit D at rnd_norm.vhd.
[74] *See* Exhibit D at rnd_norm.vhd:61-73.
[75] *See* Exhibit D at rnd_norm.vhd:55-60, 64, 70.
[76] *See* Exhibit D at rnd_norm.vhd:61-73.

104. Within the *rnd_norm* module, the sign bit of input `IN1` is simply propagated in pipelined fashion through a series of timing registers to the sign bit of output `OUT1` because normalization and rounding do not affect the sign of the floating-point input.[77]

105. The *rnd_norm* module first performs normalization of the floating-point input `IN1` using the `norm` component, which is built from the lower-level hardware module *normalizer* and performs normalization by shifting the input mantissa bits to normalized form, incrementing or decrementing the exponent accordingly, and dropping the leading integer bit (the implied '1').[78]

106. The *rnd_norm* module then performs rounding to ultimately output a result that has an overall bitwidth of m = `exp_bits` + `man_bits_out` + 1.[79] Rounding refers to the well-known process of selecting an approximation of a particular value that fits within a desired degree of precision. For example, it may be undesirable, unnecessary, or impossible to store all the bits of the number 3.14159265359 (which is itself an approximation of the number π, pi). Instead, one might *round* this number to 3.14 and store only that value due to expedience, necessity, or otherwise.

107. The module *rnd_norm* supports two modes (*i.e.*, methods) of rounding: a method commonly known as rounding to zero, and another method commonly referred to as rounding to nearest. In the first mode, rounding to zero, lower-order bits in the mantissa are simply truncated, *i.e.*, deleted, thereby reducing the mantissa bitwidth to the desired size. Truncating lower-order bits from a floating-point value can only have the effect, if any, of reducing the overall magnitude of the number (*i.e.*, bringing it closer to zero), which is why this mode of rounding is commonly referred to as rounding to zero.

108. The second rounding mode, rounding to nearest, operates more or less as its name suggests: the input is rounded to the nearest value representable within the bitwidth of the output

[77] *See* Exhibit D at rnd_norm.vhd:80-83, 101, 106, 156-158.
[78] *See* Exhibit D at normalizer.vhd.
[79] *See* Exhibit D at rnd_norm.vhd:70.

mantissa, with the input rounded to the higher representable value when it is equidistant to the two nearest representable values. The way this is implemented, essentially, is by adding '1' to the input mantissa bit that is one digit to the right of the least-significant bit of the output mantissa bitwidth (*i.e.*, to the most-significant bit that might otherwise be truncated based on the output mantissa bitwidth), then truncate to the desired bitwidth of the output mantissa. (The `ROUND` signal discussed below is used as the '1' that is used in this addition; thus, if `ROUND` is set to 0, which is for the round-to-zero mode of rounding, then the '1' is not added.) As an illustration of how this works, consider a decimal (base 10) input with one integer digit and one fraction digit, and round to nearest is being used to round to the nearest integer. This procedure would add 0.5 to the input, then truncate all digits after the radix (decimal) point. Thus, 4.1 + 0.5 = 4.6, which would be rounded via simple truncation to 4. And 4.6 + 0.5 = 5.1, which would be rounded via simple truncation to 5. For a value that presents a "tie," 4.5 + 0.5 = 5.0, which would be rounded via simple truncation to 5.

109. The rounding mode is a run-time parameter specified by the input signal `ROUND`[80] When `ROUND` is '1', rounding to nearest is used; when `ROUND` is not set (*i.e.*, the signal is '0'), rounding to zero is used. The different rounding modes in *rnd_norm* are implemented using the `rnd_add` component, which is built from the lower-level hardware module *round_add* by feeding the rounding-mode signal `ROUND` into the `rnd_add` component.[81]

C. VFLOAT System Setup and Implementation at Northeastern University

110. As discussed above, VFLOAT is a library of parameterized hardware modules for performing arithmetic operations on variable-precision floating-point numbers using FPGAs. In addition to developing VFLOAT and making it available to others as described below, we also mapped the designs to FPGA hardware in order to validate and further our work; for example, to evaluate how many complete floating point arithmetic operators could be deployed within a particular FPGA.

[80] *See* Exhibit D at rnd_norm.vhd:67, 145, 162.
[81] *See* Exhibit D at round_add.vhd; Exhibit D at rnd_norm.vhd:133-151, 161-162.

111. The specific FPGA hardware we used was the first-generation "WILDSTAR" reconfigurable computing engine, which was commercially available from Annapolis Micro, a seller of computing hardware based in Annapolis, Maryland. The WILDSTAR reconfigurable computing engine was first announced by Annapolis Micro in mid-1999 and had been available commercially for a year or two by the time of our work.[82] As noted above, Annapolis Micro produced subsequent iterations of WILDSTAR reconfigurable computing engine in later years, including, by 2008, the WILDSTAR 5, which featured 3 Xilinx Virtex-5 FPGAs.[83]

112. The particular form factor of the WILDSTAR reconfigurable computing engine that we used in our work was the PCI board. Shown below is an exemplary photo of the WILDSTAR PCI board.[84] The board's PCI bus interface, which can be seen in the below photo, was used to connect it to a host workstation.

113. The WILDSTAR board we used contained 3 Xilinx Virtex XCV1000 FPGAs, each containing the equivalent of just over 1,000,000 (1,124,022) system gates for a total of 6,144 CLBs (with each CLB comprising 2 slices for a total of 12,288 slices per FPGA).[85] The WILDSTAR board also had built-in random access memory that was accessible by the Xilinx

[82] *See* Xcell, Vol. 32 at 40 (Q2 1999), *available at* https://www.xilinx.com/publications/archives/xcell/Xcell32.pdf [LEESER000041 at 80].
[83] *See* https://web.archive.org/web/20081004190813/http://www.annapmicro.com/ws5pci.html [LEESER000120].
[84] This photo is from the slides for my presentation at HPEC 2002 in September 2002, which is discussed below.
[85] *See* Virtex™ 2.5 V Field Programmable Gate Arrays Datasheet at 1 (v1.7, Oct. 1, 1999) [XILINX-GOOG-SUB00000069 at 69]; Virtex™ 2.5 V Field Programmable Gate Arrays Datasheet at 1 (v2.5, April 2, 2001) [XILINX-GOOG-SUB00000522 at 522].

FPGAs. Specifically, the WILDSTAR board had 40 megabytes of SRAM, which was accessible to both the host workstation as well as each of the Xilinx FPGAs.

114. Our WILDSTAR board was installed in a standard Intel-based, x86 host workstation that was located in the RPL at Northeastern University. The workstation's CPU was a commercially available Intel Pentium III processor. The Intel Pentium III processor contained four IEEE standard floating-point multipliers, which were each capable of performing multiplication on IEEE single-precision floating-point values (*i.e.*, floating-point numbers in the 32-bit IEEE single-precision format).[86]

115. Once a bitstream has been generated and downloaded to one of the WILDSTAR's FPGAs, the resulting hardware or circuitry programmed onto the FPGA is activated by a "start" signal sent from the workstation, which was functionality provided by a board support package.

116. As noted above, in addition to the physical WILDSTAR board itself, Annapolis Micro provided software referred to as a board support package (BSP) that ran on the x86 host machine and supported downloading the bitstream from the host, runtime communication with the host (e.g., sending a "start" signal), and communication with on-board memory.

D. Estimating Size of VFLOAT Arithmetic Modules and Mapping Many VFLOAT Modules to FPGAs on the WILDSTAR Reconfigurable Computing Engine

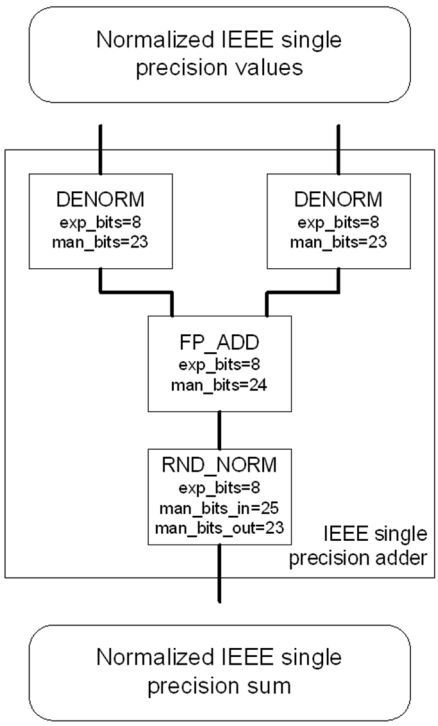
117. One area of interest in our development and implementation of VFLOAT, which stemmed from the goal of allowing greater parallelism and more efficient use of hardware resources through flexible control over signal bitwidths, was to determine how many complete VFLOAT arithmetic operators could in practice be mapped to FPGA hardware with a given number of resources. Because the size of a complete VFLOAT arithmetic operator (and thus how many could "fit" within a given FPGA) depends on the bitwidth of the floating-point format it takes as inputs and produces as output, we explored this question using a variety of different

[86] *See* Microprocessor Report (Vol. 13, No. 3, Mar. 8, 1999), *available at* https://docencia.ac.upc.edu/ETSETB/SEGPAR/microprocessors/pentium3%20(mpr).pdf [LEESER000130].

floating-point number formats supported by VFLOAT, ranging from total bitwidths of as few as 8 bits up to as many as 32 bits, including the 32-bit IEEE single-precision format. One reason for this experiment is that we desired to perform, and show to others, an "apples-to-apples" comparison of how many complete floating-point operators would fit on an FPGA using the IEEE single-precision format versus other formats with lower bitwidths.

118. Our investigation of these issues was based on assembling, implementing, and evaluating arithmetic operators that perform a complete floating-point operation, meaning operators that included both (i) denormalization (*denorm*) modules on the input side, and (ii) a rounding and normalization (*rnd_norm*) module on the output side, which I sometimes refer to in this section as "complete" arithmetic operators. For example, when determining how many complete VFLOAT floating-point addition operators using a given floating-point format would fit on an FPGA, our assessment was based on assembling and synthesizing the complete operator using two instances of an appropriately parameterized VFLOAT denormalization module *denorm* (one for each of the input operands), one instance of an appropriately parameterized VFLOAT floating-point addition module *fp_add* (to perform floating-point addition), and one instance of an appropriately parameterized VFLOAT rounding and normalization module *rnd_norm* (to perform rounding and normalization on the output of *fp_add*). For example, as part of our analysis, we assembled and evaluated complete VFLOAT floating-point addition operators for the IEEE single-precision floating-point format (1 sign bit, 8 exponent bits, and 23 mantissa bits) using two *denorm* modules, one *fp_add* module, and one *rnd_norm* module. This is illustrated in the following diagram,[87] which shows how we built an IEEE single-precision adder using VFLOAT modules as just described:

[87] This diagram is taken from the slides for my presentation at HPEC 2002 (which is discussed in more detail below).



119. With respect to the complete VFLOAT floating-point addition operator for the IEEE single-precision floating-point format, we determined that 31 such operators would fit on just one of the three Xilinx Virtex XCV1000 FPGAs present on the WILDSTAR reconfigurable computing engine, and that a single complete VFLOAT IEEE single-precision addition operator used 305 slices of the FPGA.[88]

120. We also evaluated this issue for other floating-point formats besides IEEE single-precision, including the number of such operators that could fit on a single Xilinx Virtex XCV1000 FPGA. We synthesized complete addition and multiplication operators for a variety of floating-point formats with total bitwidths of 8, 12, 16, 24, and 32 bits to determine how many would fit on a single Xilinx Virtex XCV1000 FPGA. These included, for example, a 16-bit

[88] This is corroborated, for example, by the slides for my presentation at HPEC 2002 (discussed in more detail below), at which I discussed how our work with VFLOAT demonstrated that 31 adders for the IEEE single-precision format would fit on a Xilinx VirtexXCV1000 FPGA. *See* Exhibit C at 5, 19. It is also corroborated by the abstract that I co-authored with Mr. Belanović for presentation at FPL 2002, which is discussed further below. *See* P. Belanović & M. Leeser, *A Library of Parameterized Floating Point Modules and Their Use* at 663-64 (Springer-Verlag 2002) ("The total design mapped to a Xilinx Virtex 1000 takes up 305 slices, or just under 2.5% of the chip.") [GOOG-SING-00020145 at 20151-52].

format with 6 exponent bits and 9 fraction bits, which we denominated the "C2" format; a 16-bit format with 5 exponent bits and 10 fraction bits, which we denominated the "C1" format; and a 16-bit format with 4 exponent bits and 11 fraction bits, which we denominated the "C0" format.[89] For all of the formats we considered, we implemented the complete VFLOAT floating-point addition and multiplication operators, as described above (*i.e.*, including the denormalization and rounding/normalization modules), in FPGA hardware and determined the maximum number of such operators that could in practice be mapped to a Xilinx Virtex XCV1000 FPGA. In order to ensure that our determinations regarding the maximum number of complete VFLOAT floating-point addition and multiplication operators that could fit on a Xilinx Virtex XCV1000 FPGA were realistic, we assumed that a certain percentage of resources on the FPGA would need to be reserved for routing overhead, and thus would be unavailable. Specifically, we conservatively assumed that approximately 15% of FPGA resources would need to be reserved for routing overhead, leaving only about 85% of the FPGA for floating-point operators.

121. For the C2 format (16-bit floating-point format with 6 exponent bits and 9 fraction bits), for example, we determined that 81 complete floating-point addition operators would fit on a single Xilinx Virtex XCV1000 FPGA, and 61 complete floating-point multiplication operators would fit on a single Xilinx Virtex XCV1000 FPGA. As a further example, for the C1 format (16-bit floating-point format with 5 exponent bits and 10 fraction bits), we determined that 65 complete floating-point addition operators would fit on a single Xilinx Virtex XCV1000 FPGA, and 51 complete floating-point multiplication operators would fit on a single Xilinx Virtex XCV1000 FPGA. As an additional example, for the C0 format (16-bit floating-point format with 4 exponent bits and 11 fraction bits), we determined that 76

[89] This is corroborated, for example, by Mr. Belanović's written thesis. *See* P. Belanović, *Library of Parameterized Hardware Modules for Floating-Point Arithmetic with An Example Application* at 46-47 (Northeastern Univ. 2002) (listing multiple 16-bit floating-point formats used in synthesis work with VFLOAT, including "C0" format with 4-bit exponent and 11-bit fraction, "C1" format with 5-bit exponent and 10-bit fraction, and "C2" format with 6-bit exponent and 9-bit fraction) [GOOG-SING-00020062 at 20107-108].

complete floating-point addition operators would fit on a single Xilinx Virtex XCV1000 FPGA, and 44 complete floating-point multiplication operators would fit on a single Xilinx Virtex XCV1000 FPGA.[90]

122. To confirm and validate our conclusions, for each floating-point format that we considered, we actually synthesized the maximum number of complete VFLOAT floating-point multiplication and addition operators that we determined would fit on a single Xilinx Virtex XCV1000 and mapped that number of operators to the FPGA. For example, for each floating-point number format under consideration, we synthesized the maximum number of complete VFLOAT floating-point multiplication operators that we determined would fit on a single Xilinx Virtex XCV1000, and mapped that number of complete operators to one of the Xilinx Virtex XCV1000 FPGAs on our WILDSTAR board by generating a bitstream using commercially available FPGA software tools (such as Xilinx Alliance tools and Synplicity Pro). We then loaded the bitstream onto the Xilinx Virtex XCV1000 (using the BSP from Annapolis Micro), which resulted in circuits in the FPGA hardware embodying that number of complete VFLOAT floating-point multiplication operators for the given floating-point format. The same is true for complete VFLOAT floating-point addition operators.

123. For example, relative to the "C2" format discussed above, we synthesized a design that contained 61 complete VFLOAT floating-point multiplication operators—which, as explained above, we determined was the maximum number that would fit on one of the three Xilinx Virtex XCV1000 FPGAs in our WILDSTAR board—and loaded the corresponding bitstream onto a Xilinx Virtex XCV1000 to implement that many complete VFLOAT floating-point multiplication operators in hardware. Similarly, we also synthesized a design that contained 81 complete VFLOAT floating-point addition operators using the "C2" format—which, as explained above, we determined was the maximum number that would fit on one of the

[90] These conclusions regarding the C0, C1, and C2 formats are corroborated, for example, by Mr. Belanović's written thesis. *See* P. Belanović, *Library of Parameterized Hardware Modules for Floating-Point Arithmetic with An Example Application* at 47 (Northeastern Univ. 2002) [GOOG-SING-00020062 at 20107].

three Xilinx Virtex XCV1000 FPGAs in our WILDSTAR board—and loaded the corresponding bitstream onto a Xilinx Virtex XCV1000 to implement that many complete VFLOAT floating-point addition operators in hardware.

E. Public Disclosures Relating to VFLOAT and Our System Setup

124. The original code comprising VFLOAT, along with documentation describing the purpose and functionality of the library, was made publicly available via a dedicated webpage on the website for Northeastern University's Department of Electrical and Computer Engineering around the time the initial version was completed (*i.e.*, in approximately 2002), and has remained publicly available since then. For example, the Internet Archive's Wayback Machine maintains a true and correct copy of the website for VFLOAT as it existed in early March 2003,[91] a copy of which is attached to my report as **Exhibit E**. The current webpage for VFLOAT is https://coe.northeastern.edu/Research/rcl/projects/floatingpoint/index.html [LEESER000146] and contains a link to the original code developed in 2002. That code can still be downloaded and, as noted earlier, is attached in full to my report as **Exhibit D**.

1. Mr. Belanović's May 2002 Oral Thesis Defense

125. In addition to making the code for VFLOAT publicly available via a dedicated webpage as described above, I and others presented VFLOAT and our related work through various presentations, conferences, workshops, meetings, and the like. For example, in May 2002, Mr. Belanović gave an oral presentation in connection with defending his thesis, which was titled "Library of Parameterized Modules for Floating-Point Arithmetic with An Example Application." Specifically, Mr. Belanović's thesis defense took place in a presentation at 10:00 a.m. on Wednesday, May 8, 2002, in Room 206 of the Egan Research Center, which is on the Northeastern University campus at 360 Huntington Ave., Boston, Massachusetts.

[91] *See* https://web.archive.org/web/20030313114351/http://www.ece.neu.edu:80/groups/rpl/projects/floatingpoint/index.html.

126. As is typical for a thesis defense, Mr. Belanović's presentation to the thesis committee was public, and he was later awarded his M.S. degree in Electrical Engineering from the Northeastern University Department of Electrical and Computer Engineering on the basis of the presentation and underlying work. Prior to Mr. Belanović's oral presentation of his thesis, an email was distributed to the faculty and students of the Department of Electrical Engineering announcing his thesis defense presentation and inviting any interested members of the community to attend. The substance of that email announcement, which I saved by way of an email forwarded to myself in approximately 2005, is attached as **Exhibit F**.

127. As noted, Mr. Belanović's thesis presentation was public, meaning that any members of the Northeastern University community or the general public were welcome; there were no restrictions on who was permitted to attend Mr. Belanović's thesis defense, nor were attendees subject to any restrictions on subsequent disclosure or discussion of the presentation. As Mr. Belanović's advisor and a member of his thesis committee, I attended Mr. Belanović's thesis defense in May 2002, as did other members of the committee including Dana Brooks and Waleed Meleis.[92] Other graduate students I supervised at the time attended the thesis defense as well, including Wang Chen, Haiqian Yu, and Heather Quinn, as it was my practice to invite all my graduate students and suggest that they attend the thesis defense of any graduate student whom I supervised. Mr. Belanović's oral thesis defense presentation would have lasted approximately 1 hour—divided between approximately 45 minutes of presentation by Mr. Belanović followed by approximately 15 minutes for a question-and-answer session—which was the typical amount of time allotted for a thesis defense at the Department of Electrical and Computer Engineering.

128. During his thesis defense, Mr. Belanović orally presented on a number of aspects of VFLOAT and our work related to VFLOAT, including displaying and discussing results tables, such as Table 2.2 and Figures 2.10 and 2.11 from his written thesis, relating to the

[92] The members of Dr. Belanović's thesis committee are identified in the first few pages of his written thesis. *See* GOOG-SING-00020062 at 20063-64.

conclusions we reached regarding the maximum number of complete VFLOAT floating-point arithmetic operators that would fit on a single one of the three Xilinx Virtex XCV1000 FPGAs on the WILDSTAR board that we used.

2. Other Disclosures at Northeastern University

129. Other members of the Northeastern University community also learned of and became familiar with VFLOAT during the time it was being developed and in subsequent years after it was made publicly available, including for example the graduate students who contributed to later iterations or revisions of VFLOAT over the years (*e.g.*, Haiqian Yu in 2003 and Xiaojun Wang in 2008) as well as several other graduate students whom I supervised, including Shawn Miller, Joshua Noseworthy, Albert A. Conti III, and Ben Cordes. The RPL itself, where our physical workstation described above was located, was a shared space in which, at any given time, at least a dozen and possibly as many as 20 other students and faculty members within the Department of Electrical and Computer Engineering routinely worked and to which they had unrestricted access. In addition, I specifically recall demonstrating VFLOAT and our physical workstation that included the WILDSTAR reconfigurable computing engine in approximately 2002 to Laurie Smith King, a Professor of Computer Science at Holy Cross University, who spent a sabbatical at Northeastern University during that timeframe.

3. HPEC 2002

130. In addition, during my secondary appointment as a Visiting Scientist at MIT's Lincoln Laboratories from June to December 2002 (discussed above), I attended and presented at the Sixth Annual Workshop on High Performance Embedded Computing (HPEC 2002), which was held at the MIT Lincoln Laboratory September 24-26, 2002. HPEC was intended as a forum where researchers from academia, industry, and government can discuss techniques, approaches, and ongoing developments relevant to high-performance real-time computing. A copy of the agenda for HPEC 2002 is attached as **Exhibit G**. In advance of the workshop, I authored (along with Mr. Belanović) a 2-page abstract with a summary of our proposed presentation (the "HPEC

2002 Abstract"), which was titled "A Library of Parameterized Hardware Modules for Floating-Point Arithmetic and Their Use" and was submitted to (and accepted by) the workshop organizers for presentation at HPEC 2002. The HPEC 2002 Abstract is attached as **Exhibit H**.

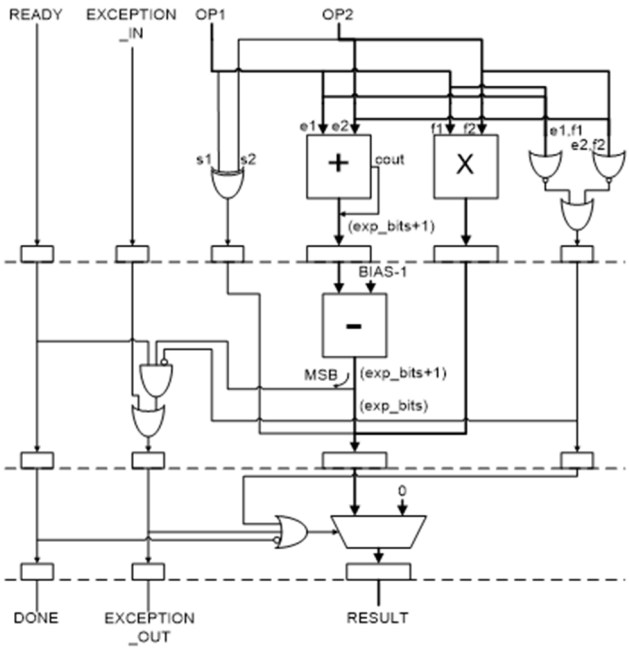
131. As reflected in the agenda for HPEC 2002, on the first day of the workshop, Tuesday, September 24, 2002, I presented various aspects of our work related to VFLOAT in a session entitled "A Library of Parameterized Hardware Modules for Floating-Point Arithmetic and Its Use."[93] Approximately 100 individuals attended HPEC 2002 overall, and approximately 30-40 individuals attended the session in which I presented various aspects of VFLOAT. The slides accompanying my oral presentation at HPEC 2002 are attached as **Exhibit C**. As reflected in those slides, I presented on a number of different aspects of VFLOAT, our related work, and applications thereof.

132. As reflected in my HPEC 2002 slides, I presented and discussed certain hardware details of the workstation we used in the RPL at Northeastern University for implementing VFLOAT.[94] For example, I explained that our system setup used a WILDSTAR reconfigurable computing engine from Annapolis Micro and that the WILDSTAR board had three Xilinx Virtex XCV1000 FPGAs.[95] I also discussed that we had developed VFLOAT in VHDL and mapped VFLOAT arithmetic modules to the Xilinx FPGAs.[96]

133. Similarly, during my presentation at HPEC 2002, I discussed the various modules that comprised VFLOAT, including specifically discussing and explaining the design (*i.e.*, structure and behavior) of the exponent addition and mantissa multiplication module (*fp_mul*), the denormalization module (*denorm*), and the normalization and rounding module (*rnd_norm*).[97] For example, I disclosed and explained that, as part of a complete floating-point multiplication operation, *fp_mul* was the VFLOAT arithmetic module that performed exponent

[93] *See* Exhibit G (HPEC 2002 Agenda).
[94] *See* Exhibit C at 17 (HPEC 2002 Slides).
[95] *See* Exhibit C at 17 (HPEC 2002 Slides).
[96] *See* Exhibit C at 17 (HPEC 2002 Slides).
[97] *See* Exhibit C at 8, 11-16 (HPEC 2002 Slides).

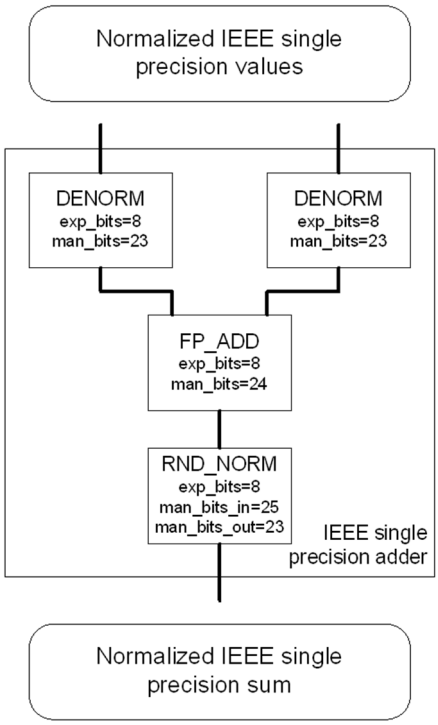
addition and mantissa multiplication, *denorm* was the VFLOAT module that "insert[s] [the] implied digit" into a normalized floating-point representation of a number, and *rnd_norm* was the VFLOAT module that "[r]eturns [its] input to normalized format."[98] In addition, I specifically discussed and disclosed the structure of the circuits that result from implementing each of the various VFLOAT floating-point arithmetic modules in hardware. For example, I discussed and disclosed to attendees the structure of the circuit that results from implementing the module *fp_mul* in hardware, using the following diagram:[99]



134. I also discussed how to assemble various VFLOAT modules to create a complete circuit for performing floating-point arithmetic operations such as addition and multiplication. For example, I specifically explained how to assemble a complete floating-point operator in hardware using VFLOAT by combining and connecting two *denorm* modules, one *fp_add* module, and one *rnd_norm* module.[100] In addition, for example, I discussed and disclosed how the *rnd_norm* module, due to its function of performing normalization and rounding, was

[98] *See* Exhibit C at 11-12, 14 (HPEC 2002 Slides).
[99] *See* Exhibit C at 14 (HPEC 2002 Slides).
[100] *See* Exhibit C at 10 (HPEC 2002 Slides).

"[d]esigned to follow arithmetic operation(s)."[101] This discussion and disclosure is also illustrated, for example, in the following diagram that I included in the slides for my HPEC 2002 presentation:[102]



135. During my presentation at HPEC 2002, I further discussed and disclosed our synthesis results and conclusions regarding the number of complete floating-point arithmetic operators that would fit on the hardware we used. This was an important part of my presentation because one of the principal motivations I identified in the presentation was to exploit parallelism to accelerate applications.[103] In particular, I compared the number of circuits embodying complete floating-point arithmetic operators of relatively lower precision that could be deployed in parallel on an FPGA to the number of such circuits that could be deployed in parallel if using IEEE single-precision formats.[104] For example, I disclosed and discussed our results showing that 85 complete floating-point multipliers using a 12-bit format could fit on a

[101] *See* Exhibit C at 12 (HPEC 2002 Slides).
[102] *See* Exhibit C at 10 (HPEC 2002 Slides).
[103] *See* Exhibit C at 3, 18-19 (HPEC 2002 Slides).
[104] *See* Exhibit C at 5, 18-19 (HPEC 2002 Slides).

single Xilinx Virtex XCV1000 FPGA whereas only 13 complete floating-point multipliers could fit on the same FPGA if using the IEEE single-precision format (with 32 bits).[105]

136. I also specifically discussed and disclosed during my HPEC 2002 presentation our conclusions regarding the median number of complete floating-point arithmetic operators that would fit on a single Xilinx Virtex XCV1000 FPGA, and further discussed and disclosed examples of specific floating-point number formats for each bitwidth.[106] For example, I specifically discussed and disclosed that at least 50 complete floating-point multipliers built from VFLOAT modules and using a 16-bit floating-point format would fit on a single Xilinx Virtex XCV1000 FPGA.[107]

4. Disclosures to / at Los Alamos National Laboratory

137. Further public disclosures related to VFLOAT took place in the context of meetings with, and presentations to, officials at LANL. As discussed above, VFLOAT arose out of and became the deliverable for our work for LANL under the sub-contract discussed above (*i.e.*, "Acceleration of Scene Classification and Spectral Unmixing with Reconfigurable Computing," as identified in my CV). In addition to actually delivering VFLOAT to LANL as part of the work described above, I personally traveled to New Mexico on several occasions in the 2002 timeframe to present our work to LANL, including presenting and describing various custom floating-point number formats that could be used with VFLOAT as well as our synthesis results such as the number of arithmetic cores that could be implemented within a single FPGA unit using various custom floating-point number formats. During those presentations to officials at LANL, we presented on and discussed specific floating-point number formats that may be suitable for use with VFLOAT, including formats with bitwidths smaller than IEEE standard formats, such as a format with a 6-bit biased exponent and 9-bit fraction field. Mr. Belanović also traveled with me to LANL on at least one occasion. During those meetings, we presented

[105] *See* Exhibit C at 5 (HPEC 2002 Slides).
[106] *See* Exhibit C at 18-19 (HPEC 2002 Slides).
[107] *See* Exhibit C at 19 (HPEC 2002 Slides).

our work to officials in the Space and Remote Sensing Sciences Group at LANL, and our presentations to and discussions with them were neither classified nor covered by any type of non-disclosure agreement or obligation.

5. Other Disclosures

138. In September 2002, Mr. Belanović orally presented an overview of VFLOAT and our application of VFLOAT at the 12th International Conference on Field Programmable Logic and Application (FPL 2002). Specifically, Mr. Belanović presented an overview of VFLOAT as well as our particular application of VFLOAT during an afternoon session on Tuesday, September 3, 2002, that was focused on FPGA-based arithmetic.[108] In advance of the conference, Mr. Belanović and I co-authored a paper (the "FPL 2002 Abstract") summarizing the proposed presentation, which summary was titled "A Library of Parameterized Floating Point Modules and Their Use" and was submitted to (and accepted by) the conference organizers for presentation at FPL 2002.[109] The FPL 2002 Abstract was published in the conference proceedings for FPL 2002.[110]

139. At the 7th annual Military and Aerospace Programmable Logic Devices International Conference (MAPLD '04), which was held in September 2004 in Washington, D.C., Xiaojun Wang and I presented an overview of our work related to VFLOAT, including the use of VFLOAT on then-current state of the art FPGAs (specifically, the Xilinx Virtex II XC2V3000 FPGA on a subsequent generation of WILDSTAR hardware from Annapolis Micro), the specific application of VFLOAT to K-means clustering for multispectral satellite images, and our continued work with VFLOAT since it was originally developed. For example, we discussed and disclosed to attendees the various arithmetic modules comprising VFLOAT, including *fp_mul*, *denorm*, and *rnd_norm*. We also described and disclosed our then-recent addition of

[108] *See* https://www.lirmm.fr/fpl02/Prog.html [LEESER000137].
[109] I understand that a copy of the FPL 2002 Abstract was produced in this litigation at GOOG-SING-00020145.
[110] Specifically, the FPL 2002 Abstract was published by Springer-Verlag as part of its Lecture Notes in Computer Science (LNCS) series. *See* https://link.springer.com/chapter/10.1007/3-540-46117-5_68 [LEESER000151].

modules for performing floating-point division (*fp_div*) and floating-point square root (*fp_sqrt*) to VFLOAT. Approximately 100 people attended the MAPLD '04 conference. Our presentation slides for MAPLD '04 are attached to my report as **Exhibit I**. In advance of the conference, Ms. Wang, Haiqian Yu, and I co-authored a paper (the "MAPLD '04 Abstract") summarizing our proposed presentation, which was titled "A Parameterized Floating-Point Library Applied to Multispectral Image Clustering" and was submitted to (and accepted by) the conference organizers for presentation at MAPLD '04. A copy of the MAPLD '04 Abstract is attached to my report as **Exhibit J**.

VII. VALIDITY ANALYSIS OF THE ASSERTED CLAIMS

140. As explained above, I understand that the Asserted Claims in this case are claim 53 of the '273 patent and claim 7 of the '156 patent.

141. Below, I detail my opinions regarding whether the Asserted Claims are invalid as anticipated and/or obvious.

A. '273 Patent, Claim 53

142. Claim 53 of the '273 patent depends from claim 43, which in turn depends from independent claim 36. Including the limitations from claims 36 and 43, claim 53 reads:

> **36.** A device:
>
> comprising at least one first low precision high-dynamic range (LPHDR) execution unit adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value,
>
> wherein the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/65,000 through 65,000 and for at least X=5% of the possible valid inputs to the first operation, the statistical mean, over repeated execution of the first operation on each specific input from the at least X % of the possible valid inputs to the first operation, of the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input differs by at least Y=0.05% from the result of an exact mathematical calculation of the first operation on the numerical values of that same input;
>
> wherein the number of LPHDR execution units in the device exceeds the non-negative integer number of execution units in the device adapted to

execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide.

43. The device of claim 36, wherein the number of LPHDR execution units in the device exceeds by at least one hundred the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide.

53. The device of claim 43, wherein the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/1,000,000 through 1,000,000.

143. For the reasons explained below, it is my opinion that claim 53 of the '273 patent is invalid as anticipated due to the claimed invention being known or used by others in this country before the priority date of the Asserted Patents, being in public use in this country more than one year prior to the priority date for the Asserted Patents, and/or being previously made in this country by another inventor who had not abandoned, suppressed, or concealed the invention, based on the creation, public availability, public disclosure, public knowledge, and public use of and regarding VFLOAT and its use on FPGA hardware. In addition, it is my further opinion that, even if claim 53 of the '273 patent is not anticipated, the claimed invention would have been obvious to a person having ordinary skill in the art at the time based on the creation, public availability, public disclosure, public knowledge, and public use of and regarding VFLOAT in combination with FPGA hardware available at the time of the claimed invention.

1. "A device"

144. To the extent that this preamble to claim 53 is limiting, it is my opinion that "[a] device," as recited in the context of the claimed invention, was publicly known and/or used by others in this country before the priority date of the Asserted Patents, in public use in this country more than one year prior to the priority date for the Asserted Patents, and previously made in this country by another inventor who had not abandoned, suppressed, or concealed the invention by virtue of the creation, public availability, public disclosure, public knowledge, and public use of and regarding VFLOAT and its use on FPGA hardware.

145. In my opinion, a workstation containing an FPGA-based reconfigurable computing engine, such as a WILDSTAR board from Annapolis Micro, constitutes a "device." By approximately 2002, at least myself and Mr. Belanović had known and used such a workstation, *i.e.*, the workstation located in the RCL at Northeastern University (then known as the Rapid Prototyping Laboratory) that contained our WILDSTAR PCI board

146. We had also publicly disclosed such knowledge and use. I incorporate by reference my discussion above regarding public disclosures of and relating to VFLOAT. As just one example of such public disclosure regarding knowledge and use of a "device" as claimed in claim 53, I discussed and disclosed to the several dozen attendees of my presentation at HPEC 2002 (discussed in more detail above) certain hardware details of the workstation we used in the RPL at Northeastern University for developing and implementing VFLOAT. Among other things, I discussed and disclosed that our system setup used a WILDSTAR reconfigurable computing engine from Annapolis Micro and that the WILDSTAR board had three Xilinx Virtex XCV1000 FPGAs, even showing the attendees a photo of the WILDSTAR board. I also discussed and disclosed that we had developed VFLOAT in VHDL and mapped VFLOAT arithmetic modules to the Xilinx FPGAs on the WILDSTAR board. My attendance and presentation at HPEC 2002 is corroborated by the conference agenda[111] as well as the HPEC 2002 Abstract,[112] and these disclosures are corroborated, for example, by the slides for my HPEC 2002 presentation.[113]

[111] *See* Exhibit G (HPEC 2002 Agenda).
[112] *See* Exhibit H (HPEC 2002 Abstract).
[113] *See* Exhibit C at 17 (HPEC 2002 Slides).

2. "comprising at least one first low precision high-dynamic range (LPHDR) execution unit adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value"

147. As discussed above, I understand that the Court construed "low precision and high dynamic range" in the context of the Asserted Claims to have the meaning defined elsewhere in the claim language itself.[114]

148. More specifically, as explained above, I understand that the Court's Claim Construction Order construed the the term "low precision" in the context of the Asserted Claims to be defined by the following claim language, which appears in both claim 36 of the '273 patent (from which asserted claim 53 of the '273 patent depends) and claim 1 of the '156 patent (from which asserted claim 7 of the '156 patent depends) and which I sometimes refer to in this report as the "minimum error" limitation:[115]

> for at least X=5% of the possible valid inputs to the first operation, the statistical mean, over repeated execution of the first operation on each specific input from the at least X% of the possible valid inputs to the first operation, of the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input differs by at least Y=0.05% from the result of an exact mathematical calculation of the first operation on the numerical values of that same input;[116]

149. I further understand, as explained above, that the Court's Claim Construction Order construed the term "high dynamic range" in the context of the Asserted Claims to be defined by the following claim language, which appears in both asserted claim 53 of the '273 patent and asserted claim 7 of the '156 patent and which I sometimes refer to in this report as the

[114] Dkt. 354, Mem. and Order on Claim Construction at 16-17 ("Therefore, the Court will construe the term 'low precision and high dynamic range' as defined in the claim itself.").
[115] Dkt. 354, Mem. and Order on Claim Construction at 16-17 ("Singular contends that the term 'low precision and high dynamic range' requires no construction because the term is defined in the claim. . . . According to Singular, the term 'low precision' is defined by the following claim language: . . . .").
[116] Dkt. 354, Mem. and Order on Claim Construction at 16-17 (quoting claim 36 of the '273 patent).

"dynamic range" limitation:[117] "the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/1,000,000 through 1,000,000."[118]

150. As also discussed above, I understand that the Court construed "execution unit" to mean "processing element comprising an arithmetic circuit paired with a memory circuit."[119]

151. As further discussed above, I also understand that the Court construed "first input signal representing a numerical value" according to its plain and ordinary meaning.[120]

152. As reflected below, I have applied the Court's constructions of these terms in reaching my opinions as set forth in this report.

153. For the reasons explained below, it is my opinion that a device comprising at least one first low precision high-dynamic range (LPHDR) execution unit (*i.e.*, processing element comprising an arithmetic circuit paired with a memory circuit) adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value, as recited in the context of the claimed invention, was publicly known and/or used by others in this country before the priority date of the Asserted Patents.

154. For the reasons explained below, it is also my opinion that a device comprising at least one first low precision high-dynamic range (LPHDR) execution unit (*i.e.*, processing element comprising an arithmetic circuit paired with a memory circuit) adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value, as recited in the context of the claimed invention, was in public use in this country more than one year before the priority date of the Asserted Patents.

---

[117] Dkt. 354, Mem. and Order on Claim Construction at 17 ("[Singular] further contends that the term 'high dynamic range' is defined by the claims as 'the dynamic range of possible valid inputs to the first operation is at least as wide from 1/1,000,000 through 1,000,000 . . . .'").
[118] Dkt. 354, Mem. and Order on Claim Construction at 17 (quoting asserted claim 53 of the '273 patent).
[119] Dkt. 354, Mem. and Order on Claim Construction at 25.
[120] Dkt. 354, Mem. and Order on Claim Construction at 30.

155. For the reasons explained below, it is further my opinion that a device comprising at least one first low precision high-dynamic range (LPHDR) execution unit (*i.e.*, processing element comprising an arithmetic circuit paired with a memory circuit) adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value, as recited in the context of the claimed invention, was made in this country by another inventor who had not abandoned, suppressed, or concealed the invention.

156. By approximately 2002, at least myself and Mr. Belanović created and used VFLOAT to implement operators for performing floating-point arithmetic on FPGA hardware using both conventional number formats and number formats with custom bitwidths, as well as disclosed such knowledge and use to the individuals who attended the various conferences, presentations, or other proceedings discussed above from approximately the same timeframe.

157. For example, we used VFLOAT to implement complete floating-point multiplication operators in FPGA hardware using a variety of number formats. As discussed above, each of these complete floating-point multiplication operators combined and connected two appropriately parameterized instances of the VFLOAT denormalization module (*denorm*) to perform denormalization on the inputs to the multiplication operator, one appropriately parameterized instance of the VFLOAT exponent addition and mantissa multiplication module (*fp_mul*), and one appropriately parameterized instance of the VFLOAT rounding and normalization module (*rnd_norm*) to create a rounded and normalized output.

158. We also implemented these complete floating-point multiplication operators in FPGA hardware, using VFLOAT hardware modules as described above, for a variety of floating-point number formats. For example, we implemented a number of complete floating-point multiplication operators in FPGA hardware using VFLOAT hardware modules that performed multiplication using the 32-bit IEEE single-precision format (1 sign bit, 8 exponent bits, and 23 mantissa bits). As further discussed above, we also implemented a number of complete floating-point multiplication operators in FPGA hardware, using VFLOAT hardware modules, that

operated on several custom 16-bit floating-point formats, including the C0 format (1 sign bit, 4 exponent bits, and 11 fraction bits), the C1 format (1 sign bit, 5 exponent bits, and 10 fraction bits), and the C2 format (1 sign bit, 6 exponent bits, and 9 fraction bits). For purposes of this discussion, I refer to the complete floating-point multiplication operator created for the C2 format from VFLOAT hardware modules, as described above, and synthesized and loaded onto our WILDSTAR board (*i.e.*, one of the three Virtex XCV1000 FPGAs) as a "C2 Multiplier."

159. I turn first to the portion of the claim language that recites "**at least one first low precision high-dynamic range (LPHDR) execution unit**." As noted above, I understand that the Court, in its Claim Construction Order, construed "low precision and high dynamic range" in the context of the Asserted Claims to have the meaning defined elsewhere in the claim language itself.[121] Because the meaning ascribed by the Court to the term "low precision and high dynamic range (LPHDR)" is expressly set forth in other limitations appearing in the Asserted Claims (namely, the "minimum error" and "dynamic range" limitations), I incorporate by reference my analysis and opinions for those limitations as set forth below, and in particular how a C2 Multiplier satisfies the "minimum error" and "dynamic range" limitations. For those reasons, it is my opinion that a C2 Multiplier, when implemented in FPGA hardware (as we did), satisfies the claim language regarding "low precision high-dynamic range (LPHDR)," and in particular the Court's construction of that claim language.

160. In addition, it is my opinion that a C2 Multiplier constitutes the claimed "**execution unit**" (*i.e.*, "processing element comprising an arithmetic circuit paired with a memory circuit"). As noted above, I understand that the Court, in its Claim Construction Order, construed "execution unit" to mean "processing element comprising an arithmetic circuit paired with a memory circuit."[122] In my opinion, a person of ordinary skill in the art would have

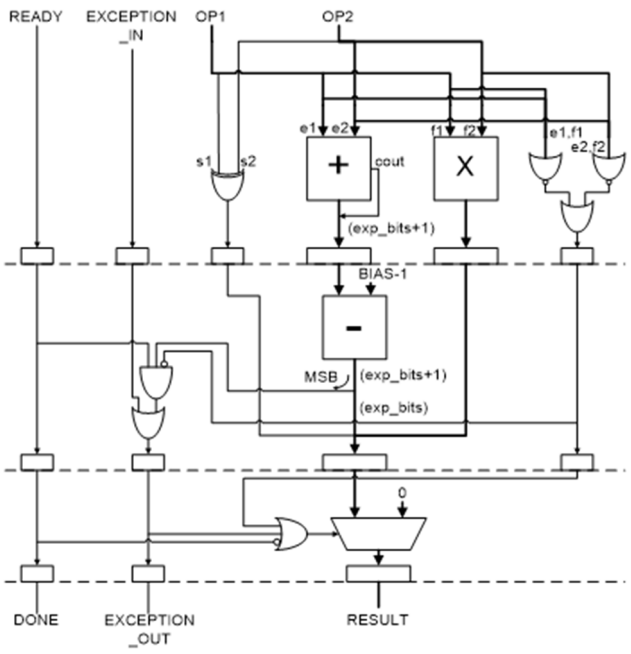
[121] Dkt. 354, Mem. and Order on Claim Construction at 16-17 ("Therefore, the Court will construe the term 'low precision and high dynamic range' as defined in the claim itself.").
[122] Dkt. 354, Mem. and Order on Claim Construction at 25.

understood that a C2 Multiplier constitutes a "processing element comprising an arithmetic circuit paired with a memory circuit."

161. To begin with, it is my opinion that the C2 Multiplier constitutes a "processing element." As discussed above, a C2 Multiplier—like other complete VFLOAT floating-point multiplication operators—is constructed from and comprises two instances of the VFLOAT denormalization module (*denorm*), one instance of the VFLOAT exponent addition and mantissa multiplication module (*fp_mul*), and one instance of the VFLOAT rounding and normalization module (*rnd_norm*). Taken together, these modules, when implemented in FPGA hardware (as we did using the WILDSTAR reconfigurable computing engine), perform the processing needed to accept two floating-point numbers as inputs, denormalize the inputs by adding back the hidden bit, perform the subsidiary arithmetic steps needed to multiply the two floating-point numbers, and perform rounding and normalization to produce a normalized output.

162. Moreover, when synthesized and loaded onto the WILDSTAR board, as we did using Synplicity Pro and Xilinx Alliance tools as well as the BSP from Annapolis Micro, the VFLOAT exponent addition and mantissa multiplication module (*fp_mul*) created an arithmetic circuit. The structure of a circuit that results from the implementation of a VFLOAT floating-point multiplier module in FPGA hardware is illustrated in the below diagram,[123] which I discussed above in my overview of the VFLOAT *fp_mul* module:

[123] *See* Exhibit C at 14 (HPEC 2002 Slides).



163. In my opinion, the circuit created when implementing *fp_mul*, as illustrated above, constitutes or comprises the claimed "arithmetic circuit" at least because the result of implementing an *fp_mul* module in FPGA hardware is, as shown above, a circuit that is designed to perform arithmetic. The exact physical implementation of the circuitry would vary based on the bitwidth parameters used for the *fp_mul* module (*i.e.*, the parameters for the bitwidth of the exponent and mantissa—`exp_bits` and `man_bits`, respectively). For example, when implementing the *fp_mul* module in FPGA hardware as part of multiple C2 Multipliers, as we did, those parameters would be 6 bits and 10 bits, respectively, for the exponent and mantissa bitwidths (the mantissa bitwidth parameter being set to 10 bits to account for the addition of the implied '1' by the *denorm* module). However, the structure of the circuitry when implemented in hardware would nonetheless adhere to what is illustrated in the diagram above.

164. As discussed above in the overview of our system setup for VFLOAT at Northeastern University, the WILDSTAR board that we used to implement VFLOAT had 40 megabytes of on-board SRAM, which was accessible by both the host workstation as well as each of the three Xilinx Virtex XCV1000 FPGAs on the WILDSTAR board. The host workstation could, for example, send data to the memory on the WILDSTAR board, which data

could then be input directly into circuits on the FPGAs. In my opinion, the WILDSTAR board's on-board SRAM constitutes a paired memory circuit—such that the arithmetic circuit embodying *fp_mul* within a C2 Multiplier on one of the WILDSTAR board's FPGAs is "paired with a memory circuit"—because it is composed of memory to which the C2 Multiplier (the claimed "execution unit") has direct access.

165. In addition, to the extent Singular contends that non-addressable registers can constitute the claimed "memory circuit" (*i.e.*, the "memory circuit" that the claimed "arithmetic circuit" is paired with), it would be my opinion that that provides an additional and independent reason for concluding that the *fp_mul* module, when implemented on the WILDSTAR board as we did, is "paired with a memory circuit." As discussed above, the VFLOAT exponent addition and mantissa multiplication module *fp_mul* includes a number of registers. These registers can be seen, for example, in a number of places in the above diagram of an *fp_mul* arithmetic circuit. As just one example, the sum of the two input exponents is stored for a short time in an intermediate timing register. While this register and the others present inside of the circuit created from *fp_mul* are not addressable in the same way as SRAM, to the extent Singular contends that non-addressable registers can constitute the claimed "memory circuit," it would be my opinion that the registers in *fp_mul* also constitute the claimed "memory circuit," such that a C2 Multiplier "compris[es] an arithmetic circuit paired with a memory circuit" in accordance with the Court's construction of "execution unit."

166. Furthermore, I understand that, in its Claim Construction Order, the Court relied on the exemplary processing element shown in Figure 4 of the shared specification of the Asserted Patents in determining the appropriate construction of "execution unit" (*i.e.*, to construe "execution unit" to mean "processing element comprising an arithmetic circuit paired with a

memory circuit").[124] I observe that the exemplary processing element shown in Figure 4 can be controlled by "control signals," as shown below:[125]

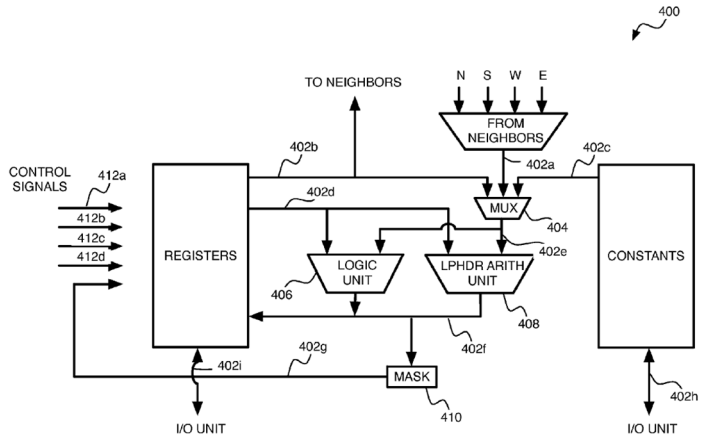


FIG. 4

167. To the extent that "processing element" under the Court's construction of "execution unit" requires control signals such as those depicted in Figure 4 of the shared specification of the Asserted Patents, it is my opinion that a C2 Multiplier comprises such control signals and thus satisfies such a requirement. For example, as discussed above in my detailed discussion of VFLOAT, the *fp_mul* module is configured to receive a "ready" signal (`READY`) that instructs it to begin processing data. The *fp_mul* module also receives, for example, an "exception" signal in (`EXCEPTION_IN`), and can produce an exception signal out (`EXCEPTION_OUT`). For further example, as also discussed above, the *rnd_norm* module receives a "ready" signal (`READY`) that instructs it to begin processing data, as well as a "round" signal (`ROUND`) that determines which rounding mode it should use. In my opinion, a person of ordinary skill in the art would have understood each one of these to constitute a control signal. Thus, in my opinion, a C2 Multiplier comprises control signals, to the extent that such are required under the Court's construction of "execution unit."

[124] Dkt. 354, Mem. and Order on Claim Construction at 22-24 ("Similarly, Figure 4 of the patent provides intrinsic evidence to the same effect[.] As described in the specification, Figure 4 depicts a processing element that pairs the LPHDR arithmetic element with memory circuits[.]").
[125] '273 patent, Fig. 4.

168. I turn next to the claim language regarding the "**first operation**" that the claimed "execution unit" is "adapted to execute," *i.e.*, that the claimed "execution unit" is "**adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value**." As discussed above, it is my opinion that a person of ordinary skill in the art would have understood that a C2 Multiplier constitutes the claimed "execution unit" (*i.e.*, "processing element comprising an arithmetic circuit paired with a memory circuit"). In my further opinion, a C2 Multiplier is "adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value" in accordance with the claim language.

169. As discussed above, a C2 Multiplier is composed of two appropriately parameterized instances of the VFLOAT denormalization module (*denorm*), one appropriately parameterized instance of the VFLOAT exponent addition and mantissa multiplication module (*fp_mul*), and one appropriately parameterized instance of the VFLOAT rounding and normalization module (*rnd_norm*). For example, as discussed above, in the case of the C2 Multiplier, the appropriate parameterization for the *fp_mul* module would be 6 bits and 10 bits, respectively, for the exponent and mantissa bitwidth parameters (`exp_bits` and `man_bits`), with the mantissa bitwidth parameter (`man_bits`) set to 10 bits in order to account for the restoration of the hidden bit by the *denorm* module, as discussed above. As discussed above in my detailed discussion of VFLOAT, the VFLOAT module *fp_mul* was designed to be used as part of floating-point multiplication (*i.e.*, multiplication of two floating-point numbers), with the two *denorm* modules and the *rnd_norm* module providing format control (*i.e.*, denormalization and rounding/normalization). Thus, the C2 Multiplier as a whole is adapted to execute floating-point multiplication, and it is my opinion that this constitutes a "first operation" in accordance with the claim language.

170. In my opinion, this "first operation" (*i.e.*, the floating-point multiplication that a C2 Multiplier is adapted to execute) is performed "on a first input signal representing a first

numerical value," in accordance with the claim language. For example, as discussed above, a C2 Multiplier includes circuits embodying two *denorm* modules that "denormalize" the inputs to the C2 Multiplier (*i.e.*, add back the hidden bit to the normalized floating-point inputs) and feed those denormalized values to inputs `OP1` and `OP2` of the circuit embodying *fp_mul* (which is the VFLOAT exponent addition and mantissa multiplication module). In my opinion, either of the floating-point number inputs to a C2 Multiplier constitutes a "a first input signal" that the "first operation" is performed on, in accordance with the claim language. As discussed above, the hardware we used for implementing VFLOAT, including to implement a number of C2 Multipliers, was a WILDSTAR board containing multiple Xilinx FPGAs. FPGAs operate on electricity and thus use electrical signals to encode and transmit data. Thus, the input to a VFLOAT *denorm* module, such as when implemented in FPGA hardware as part of a C2 Multiplier (as we did), takes the form of electrical signals and is a "first input signal" in accordance with the claim language.[126]

171. In my opinion, these electrical signals "represent[] a first numerical value" in accordance with the claim language because the input to a VFLOAT *denorm* module, such as when implemented in FPGA hardware as part of a C2 Multiplier (as we did), represents a particular real number encoded in the floating-point format. In my opinion, the real number that is encoded in a floating-point format and provided as the input to a VFLOAT *denorm* module that is implemented in FPGA hardware as part of a C2 Multiplier, is a "numerical value"—particularly when viewed in light of the Court's claim construction ruling equating signals and values in the context of the Asserted Patents.

172. In my opinion, the "first operation" I have identified (*i.e.*, the floating-point multiplication that a C2 Multiplier is adapted to execute) "produce[s] a first output signal representing a second numerical value" in accordance with the claim language. As discussed above, a C2 Multiplier includes a circuit embodying the VFLOAT rounding and normalization

[126] Indeed, in the code for the VFLOAT *denorm* module, we explicitly referred to input IN1 (which, as discussed above, is the input to that module) as an "input signal[]." *See* Exhibit D at denorm.vhd:86.

module (*rnd_norm*) to create a rounded and normalized output. As discussed above in my detailed explanation of VFLOAT, the output of *fp_mul*, which is denoted as `RESULT` in the VHDL code,[127] has a mantissa that is twice the bitwidth of the mantissa of the inputs. Normalizing that output mantissa and converting it back to the C2 format is the function of *rnd_norm*, and in particular the circuit embodying *rnd_norm* when this module is parameterized and implemented in hardware as part of a C2 Multiplier.

173. As discussed above, the output of *rnd_norm* is denoted as `OUT1` in the VHDL code and is the output of the overall C2 Multiplier when this module is implemented in hardware as part of a C2 Multiplier.[128] In my opinion, the output of the circuit embodying a *rnd_norm* module within a C2 Multiplier, which is the output of the overall C2 Multiplier, is a "first output signal representing a second numerical value" in accordance with the claim language. As discussed above, the hardware we used for implementing VFLOAT, including to implement a number of C2 Multipliers, was a WILDSTAR board containing multiple Xilinx FPGAs. As discussed above, FPGAs use electrical signals to encode and transmit data.

174. Thus, in my opinion, the output of a VFLOAT *rnd_norm* module within a C2 Multiplier, when implemented in FPGA hardware, takes the form of electrical signals and is a "first output signal" in accordance with the claim language.[129] In my opinion, these electrical signals "represent[] a second numerical value" in accordance with the claim language—namely, the particular real number that is encoded into the bits that are output from a VFLOAT *rnd_norm* module within a C2 Multiplier, when implemented in FPGA hardware. A person of ordinary skill in the art would have understood that the real number that is encoded into the bits of a floating-point format number and provided as the output of a VFLOAT *rnd_norm* module that is implemented in FPGA hardware as part of a C2 Multiplier, is a "numerical value."

---

[127] *See* Exhibit D at fp_mul.vhd:65-74.
[128] *See* Exhibit D at rnd_norm.vhd:61-73.
[129] Indeed, in the code for the VFLOAT *rnd_norm* module, we explicitly referred to the output `OUT1` (which, as discussed above, is the output of that module) as an "output signal[]." *See* Exhibit D at rnd_norm.vhd:105.

175. Accordingly, it is my opinion that each of the many C2 Multipliers that we synthesized and loaded onto a WILDSTAR board satisfy all aspects of this claim limitation—*i.e.*, that a C2 Multiplier constitutes a "first low precision high-dynamic range (LPHDR) execution unit adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value" in accordance with the claim language and the Court's claim constructions.

176. In addition, it is my opinion that a device satisfying this limitation was actually reduced to practice in this country, publicly known and used in this country, in public use in this country, and not abandoned, suppressed, or concealed. By way of example, as discussed above, to confirm and validate our conclusions regarding the maximum number of complete VFLOAT floating-point multiplication and addition operators that could fit on a single one of the three Xilinx Virtex XCV1000 FPGA on our WILDSTAR board, we synthesized and mapped that maximum number of operators to the FPGA. For the C2 format, for example, this means that we synthesized a design containing 61 C2 Multipliers and mapped that design to FPGA hardware (*i.e.*, to one of the three Xilinx Virtex XCV1000 FPGA on our WILDSTAR board), such that the FPGA contained circuitry embodying that many C2 Multipliers.

177. By way of further example, as discussed above, I personally traveled to LANL multiple times to discuss our work relating to VFLOAT with officials there, and ultimately delivered VFLOAT to LANL pursuant to the sub-contract discussed above (*i.e.*, "Acceleration of Scene Classification and Spectral Unmixing with Reconfigurable Computing," as identified in my CV). These discussions with LANL included, for example, presenting and describing various custom floating-point number formats that could be used with VFLOAT, as well as our synthesis results regarding the number of complete arithmetic operators that could be implemented within a single FPGA unit using various custom floating-point formats.

178. By way of additional example, as discussed above, during my presentation at HPEC 2002, I discussed and disclosed that our system setup used a WILDSTAR reconfigurable computing engine from Annapolis Micro and that the WILDSTAR board had three Xilinx Virtex

XCV1000 FPGAs.[130] I also discussed and disclosed that we had developed VFLOAT in VHDL and mapped VFLOAT arithmetic modules to the Xilinx FPGAs.[131] I further discussed and disclosed how to assemble VFLOAT modules to create a complete circuit for performing floating-point arithmetic operations—*i.e.*, using two *denorm* modules, one floating-point arithmetic module, and one *rnd_norm* module.[132] I also presented our synthesis results and conclusions regarding the number of complete floating-point arithmetic operators that would fit on the hardware we used, including that at least 50 complete floating-point multipliers built from VFLOAT modules and using a 16-bit floating-point format would fit on a single Xilinx Virtex XCV1000 FPGA.[133]

179. By way of still further example, as discussed above, the RPL lab itself, where our physical workstation was located, was a shared space in which many other students and faculty members of the Department of Electrical and Computer Engineering routinely worked and to which they had unrestricted access. In addition, as also explained above, many members of the Northeastern University community learned of and became familiar with VFLOAT during the time it was being developed and in subsequent years, including various graduate students as well as Laurie Smith King, a Professor of Computer Science at Holy Cross University who spent a sabbatical at Northeastern University.

180. By way of yet further example, as discussed above, at the MAPLD ’04 conference in September 2004, Xiaojun Wang and I presented an overview of our work related to VFLOAT, including our then-recent development and addition of modules for performing floating-point division (*fp_div*) and floating-point square root (*fp_sqrt*) to VFLOAT.[134]

181. Rather than enumerate here every public presentation, conference proceeding, or other public disclosure relating to VFLOAT that demonstrates how our work was publicly

[130] *See* Exhibit C at 17 (HPEC 2002 Slides).
[131] *See* Exhibit C at 17 (HPEC 2002 Slides).
[132] *See* Exhibit C at 10 (HPEC 2002 Slides).
[133] *See* Exhibit C at 3, 18-19 (HPEC 2002 Slides).
[134] *See* Exhibit J at 1 (MAPLD ’04 Abstract).

known and used in this country, in public use in this country, and not abandoned, suppressed, or concealed, I incorporate in its entirety my discussion above of the various public disclosures relating to VFLOAT, as well as the contents of those various public disclosures.

182. For these reasons, it is my opinion that a device satisfying this limitation was actually reduced to practice in this country, publicly known and used in this country, in public use in this country, and not abandoned, suppressed, or concealed.

3. "wherein the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/65,000 through 65,000"

183. For the reasons explained below, it is my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was publicly known and/or used by others in this country before the priority date of the Asserted Patents.

184. For the reasons explained below, it is also my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was in public use in this country more than one year before the priority date of the Asserted Patents.

185. For the reasons explained below, it is further my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was made in this country by another inventor who had not abandoned, suppressed, or concealed the invention.

186. For example, by approximately 2002, I and Mr. Belanović created and used VFLOAT to implement complete floating-point arithmetic operators in FPGA hardware, such as the C2 Multiplier, for which the possible valid inputs had a dynamic range at least as wide as from 1/65,000 through 65,000. As discussed above in connection with the prior limitation of this claim, it is my opinion that the floating-point multiplication that a C2 Multiplier is adapted to execute constitutes a "first operation" in accordance with the claim language. In addition, as discussed above, a C2 Multiplier implemented in FPGA hardware performs multiplication on inputs that are represented in the C2 floating-point format (*i.e.*, a 16-bit format with a 6-bit biased exponent and 9 fraction bits). It follows from simple math, was understood by at least myself and Mr. Belanović, and in my opinion would have been understood by a person of

ordinary skill in the art, that the possible valid inputs to a C2 Multiplier have a dynamic range that is wider than from 1/65,000 on the low end through 65,000 on the high end—*i.e.*, that the possible valid inputs to a C2 Multiplier include floating-point numbers that are greater than 65,000 as well as floating-point numbers that are less than 1/65,000—due to the C2 format used for the inputs to the C2 Multiplier. As discussed above in the section of my report providing technical background regarding floating-point numbers, a biased exponent that is represented using 6 bits (such as in the C2 format used for the C2 Multiplier) can range in value from at least -30 to 31,[135] such that the resulting floating-point number has a dynamic range at least as wide as from 2^{-30} (which is 1/1,073,741,824) on the low end to over 2^{31} (which is 2,147,483,648) on the high end. In my opinion, the possible valid inputs to the C2 Multiplier have such a dynamic range because both numerical inputs to the C2 Multiplier use the floating-point format we denominated "C2," which was a 16-bit format with 1 sign bit, a 6-bit biased exponent, and 9-bit fraction. In my opinion, such a dynamic range is "at least as wide as from 1/65,000 through 65,000" in accordance with the claim language.

187. In my opinion, this knowledge and use is corroborated by contemporaneous documents, including, for example, the code for VFLOAT itself. For example, as discussed above and as seen in the source code for VFLOAT, the entity declaration for the module *fp_mul* (which defines the exponent addition and mantissa multiplication module) includes a VHDL `port` declaration that defines floating-point inputs `OP1` and `OP2` to the module and a `generic` clause that parameterizes the module according to two attributes of the floating-point inputs: the bitwidth of the exponent (`exp_bits`) and the bitwidth of the mantissa (`man_bits`).[136] These parameters were meant to be customized in accordance with the designer's specific application, and, when instantiated, the module would accept (and perform multiplication on) floating-point inputs with the specified exponent and mantissa bitwidths. This is also shown, for example, in

[135] As also discussed above in the section of my report providing technical background regarding floating-point numbers, floating-point formats that follow IEEE convention reserve the smallest and largest exponents (all zeros or all ones) to represent special values.
[136] *See* Exhibit D at fp_mul.vhd:59-69.

the `port` declaration's definition of `OP1` and `OP2`, which defines those inputs based on the specified exponent and mantissa bitwidths.[137] In my opinion, the contemporaneous corroboration of this knowledge and use also includes, for example, Mr. Belanović's written thesis, which describes several example floating-point formats for use with VFLOAT, including multiple formats with a 6-bit or wider biased exponent, such as the C2 format with a 6-bit biased exponent and 9-bit fraction.[138] In my opinion, the contemporaneous corroboration of this knowledge and use further includes, for example, the FPL 2002 Abstract, which similarly describes several example floating-point formats for use with VFLOAT, including multiple formats with a 6-bit or wider biased exponent, including the C2 format.[139]

188. In addition, it is my opinion that a device satisfying this limitation was actually reduced to practice in this country, publicly known and used in this country, in public use in this country, and not abandoned, suppressed, or concealed. For example, as discussed above, to confirm and validate our conclusions regarding the maximum number of complete VFLOAT floating-point multiplication and addition operators that could fit on a single one of the three Xilinx Virtex XCV1000 FPGA on our WILDSTAR board, we synthesized and mapped that maximum number of operators to the FPGA. For the C2 format, for example, this means that we synthesized a design containing 61 C2 Multipliers and mapped that design to FPGA hardware (*i.e.*, to one of the three Xilinx Virtex XCV1000 FPGA on our WILDSTAR board), such that the FPGA contained circuitry embodying that many C2 Multipliers, as discussed above.

189. By way of further example, as discussed above, during my presentation at HPEC 2002, I discussed and disclosed VFLOAT in general, as well as how we had developed VFLOAT in VHDL and mapped VFLOAT arithmetic modules to the Xilinx FPGAs on our

[137] *See* Exhibit D at fp_mul.vhd:65-69.

[138] *See* P. Belanović, *Library of Parameterized Hardware Modules for Floating-Point Arithmetic with An Example Application* at 47 & Table 2.2 (Northeastern Univ. 2002) (listing multiple floating-point formats for use with VFLOAT including "C2" format with 6-bit exponent and 9-bit fraction) [GOOG-SING-00020062 at 20108].

[139] *See* P. Belanović & M. Leeser, *A Library of Parameterized Floating Point Modules and Their Use* at 663 (Springer-Verlag 2002) (listing multiple floating-point formats for use with VFLOAT including "C2" format with 6-bit exponent and 9-bit fraction) [GOOG-SING-00020145 at 20151].

WILDSTAR board.[140] I further discussed and disclosed how VFLOAT modules could be assembled (and in fact were assembled by us) to create a complete circuit for performing floating-point arithmetic operations—*i.e.*, using two *denorm* modules, one floating-point arithmetic module, and one *rnd_norm* module.[141] I also presented our synthesis results and conclusions regarding the number of complete floating-point arithmetic operators that would fit on the hardware we used, including our conclusion that at least 50 complete floating-point multipliers built from VFLOAT modules and using a 16-bit floating-point format (like the C2 Multiplier) would fit on a single Xilinx Virtex XCV1000 FPGA.[142]

190. By way of additional example, during his May 2002 oral thesis defense at Northeastern University, which I attended, Mr. Belanović presented on the development of VFLOAT and its use on FPGA hardware in a roughly hour-long presentation that was open to the public, as discussed above. In his oral presentation, Mr. Belanović explained that VFLOAT could be and in fact was used to implement complete floating-point arithmetic operators in FPGA hardware by combining and connecting two VFLOAT *denorm* modules, a VFLOAT arithmetic module (*e.g.*, *fp_add* or *fp_mul*), and a VFLOAT *rnd_norm* module. Mr. Belanović also specifically identified and described several exemplary floating-point number formats for use with VFLOAT arithmetic operators, including a floating-point number format comprising a 6-bit biased exponent, a 9-bit fraction, and 1-bit sign value, for a total bitwidth of 16 bits (*i.e.*, the C2 format).[143]

191. By way of still further example, as discussed above, the RPL lab itself, where our physical workstation was located, was a shared space in which many other students and faculty members of the Department of Electrical and Computer Engineering routinely worked and to

[140] *See* Exhibit C at 17 (HPEC 2002 Slides).
[141] *See* Exhibit C at 10 (HPEC 2002 Slides).
[142] *See* Exhibit C at 3, 18-19 (HPEC 2002 Slides).
[143] This is corroborated, for example, by Mr. Belanović's written thesis. *See* P. Belanović, *Library of Parameterized Hardware Modules for Floating-Point Arithmetic with An Example Application* at 47 (Northeastern Univ. 2002) (listing multiple floating-point formats for use with VFLOAT including "C2" format with 6-bit exponent and 9-bit fraction) [GOOG-SING-00020062 at 20108].

which they had unrestricted access. In addition, as also explained above, many members of the Northeastern University community learned of and became familiar with VFLOAT during the time it was being developed and in subsequent years, including various graduate students as well as Laurie Smith King, a Professor of Computer Science at Holy Cross University who spent a sabbatical at Northeastern University.

192. By way of yet further example, as discussed above, at the MAPLD '04 conference in September 2004, Xiaojun Wang and I presented an overview of our work related to VFLOAT, including our then-recent development and addition of modules for performing floating-point division (*fp_div*) and floating-point square root (*fp_sqrt*) to VFLOAT.[144]

193. Rather than enumerate here every public presentation, conference proceeding, or other public disclosure relating to VFLOAT that demonstrates how our work was publicly known and used in this country, in public use in this country, and not abandoned, suppressed, or concealed, I incorporate in its entirety my discussion above of the various public disclosures relating to VFLOAT, as well as the contents of those various public disclosures.

194. For these reasons, it is my opinion that a device satisfying this limitation was actually reduced to practice in this country, publicly known and used in this country, in public use in this country, and not abandoned, suppressed, or concealed.

4. "for at least X=5% of the possible valid inputs to the first operation, the statistical mean, over repeated execution of the first operation on each specific input from the at least X % of the possible valid inputs to the first operation, of the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input differs by at least Y=0.05% from the result of an exact mathematical calculation of the first operation on the numerical values of that same input"

195. As explained above, the C2 Multiplier that we built used a floating-point format for its inputs that included 1 sign bit, 6 exponent bits, and 9 fraction bits. That multiplier circuit was parameterized for such inputs first into the *denorm* module (for denormalization), then to the

[144] *See* Exhibit J at 1 (MAPLD '04 Abstract).

fp_mul module (for exponent addition and mantissa multiplication), and ultimately to the *rnd_norm* module (for rounding and normalization). That last step rounded the results from the *fp_mul* module, effectively removing a number of the least significant bits from the product, such that the numerical values represented by the output from the *rnd_norm* module (and therefore the C2 Multiplier of which it was a part) differed from the result of an exact mathematical calculation of the product of the multiplication operation on the numerical values of the same input.

196. An example may help illustrate the point.[145] If one were to use the C2 Multiplier (or a multiplication circuit like it), to multiply 1.111111111 by 1.222222222, the output of the complete multiplier circuit would not be the same as the output of the exact mathematical calculation. Specifically, the exact mathematical calculation would equate to 1.3580246909753100000 whereas the the *rnd_norm* module would round the number to 1.358024691 and then normalize that to 9 bits by removing the "implied 1" bit, leaving only the fractional part.

197. Notably, this loss of bits is similar to how a single-precision floating-point multiplier is generally implemented; although it starts with two 32-bit inputs, of which 23 bits of each input are the fractional part, the product of such a multiplier circuit, when following the standard, does not include all the bits of the internal mantissa multiplication in the output. Depending on the implementation (*e.g.*, truncation, rounding to nearest, etc.), the results differ to varying degrees from the results of the exact mathematical calculation on the same inputs. This is well-understood by those of skill in the art.

198. In the particular case of multiplier inputs with a floating-point format using 1 sign bit, 6 exponent bits, and 9 fraction bits, I understand from my discussions with Dr. Gustafson that, because of the reduction of fraction bits on the output of the C2 Multiplier, for at least X=5% of the possible valid inputs to the multiplication operation, the numerical values

[145] Recognizing that a computer circuit uses bits, I provide a numerical example for this illustration.

represented by the first output signal of the C2 Multiplier executing the first operation on that input differs by at least Y=0.05% from the result of an exact mathematical calculation of the first operation on the numerical values of that same input. Specifically, I understand that Dr. Gustafson used Mathematica to model the possible valid inputs to a multiplier circuit that used a floating-point format with 1 sign bit, 6 exponent bits, and 9 fraction bits and rounded the result down to 9 fraction bits, and determined that, for at least X=5% of the possible valid inputs, the statistical mean, over repeated execution of a floating-point multiplication operation on each specific input from the at least X % of the possible valid inputs to the multiplication operation, of the numerical values represented by the output signal differs by at least Y=0.05% from the result of an exact mathematical calculation of the first operation (multiplication) on the numerical values of that same input.[146] Based on our discussions, as well as Dr. Gustafson's reputation in the field, I consider it unnecessary to redo his analysis and rely on his mathematical model for purposes of this claim limitation.

199. For these reasons, it is my opinion that a device satisfying this limitation was actually reduced to practice in this country, publicly known and used in this country, in public use in this country, and not abandoned, suppressed, or concealed.

5. "wherein the number of LPHDR execution units in the device exceeds the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide"

200. For the reasons explained below, it is my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was publicly known and/or used by others in this country before the priority date of the Asserted Patents.

201. For the reasons explained below, it is also my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was in public use in this country more than one year before the priority date of the Asserted Patents.

[146] I understand that Dr. Gustafson's conclusion that the C2 Multiplier meets this limitation under either rounding mode is not dependent on whether pairs of inputs whose products will cause an overflow or underflow should be considered as "valid" inputs.

202. For the reasons explained below, it is further my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was made in this country by another inventor who had not abandoned, suppressed, or concealed the invention.

203. For example, as discussed above regarding the limitation reciting "at least one first low precision high-dynamic range (LPHDR) execution unit . . .", it is my opinion that a C2 Multiplier constitutes an "execution unit" as claimed (*i.e.*, "processing element comprising an arithmetic circuit paired with a memory circuit" in accordance with the Court's construction of "execution unit"[147]) and further satisfies the claim language regarding "low precision high-dynamic range (LPHDR)." By way of further example, as discussed above, to confirm and validate our conclusions regarding the maximum number of complete VFLOAT floating-point multiplication and addition operators that could fit on a single one of the three Xilinx Virtex XCV1000 FPGA on our WILDSTAR board, we synthesized and mapped that maximum number of operators to the FPGA. For the C2 format, for example, this means that we synthesized a design containing 61 C2 Multipliers and mapped that design to FPGA hardware (*i.e.*, to one of the three Xilinx Virtex XCV1000 FPGA on our WILDSTAR board), such that the FPGA contained circuitry embodying that many C2 Multipliers, as discussed above.

204. In addition, in my opinion, the claimed "non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide" was at most four in view of the workstation that we used to implement C2 Multipliers. As discussed above in the section describing our system setup and implementation of VFLOAT at Northeastern University, our WILDSTAR board (on which we implemented C2 Multipliers) was installed in a standard Intel-based, x86 host workstation with a commercially available Intel Pentium III processor. As also discussed above, the Intel Pentium III processor contained four IEEE standard floating-point multipliers,

[147] *See* Dkt. 354, Mem. and Order on Claim Construction at 25.

which were each capable of performing multiplication on IEEE single-precision floating-point values (*i.e.*, floating-point numbers in the 32-bit IEEE single-precision format).

205. Assuming that each of the IEEE single-precision floating-point multipliers in the Pentium III is a "processing element comprising an arithmetic circuit paired with a memory circuit" in accordance with the Court's construction of "execution unit,"[148] it is my opinion that each of these multipliers is "adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide" in accordance with the claim language because they are designed to perform floating-point multiplication on 32-bit IEEE single-precision numbers, which are "floating point numbers that are at least 32 bits wide." Moreover, four is a "non-negative integer number" in accordance with the claim language. Thus, in my opinion, the "non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide," in accordance with the claim language, is at most four based on the number of IEEE single-precision floating-point multipliers in the Pentium III.[149]

206. In light of the fact that, as discussed above, we implemented a design containing 61 C2 Multipliers in FPGA hardware (*i.e.*, to one of the three Xilinx Virtex XCV1000 FPGA on our WILDSTAR board), and the fact that there were at most four "execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide," it is my opinion that "the number of LPHDR execution units in the device exceeds the non-negative integer number of execution units in the device adapted to execute at

[148] *See* Dkt. 354, Mem. and Order on Claim Construction at 25.

[149] If the IEEE single-precision floating-point multipliers in the Pentium III are deemed not to constitute "execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide" under the Court's construction of "execution unit," then it is my alternative opinion that the maximum number of "execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide" in our workstation was zero, and my ultimate opinion regarding this claim limitation would remain unchanged, because 0 is a "non-negative integer" in accordance with the claim language and because the number 61 is greater than (*i.e.*, exceeds) the number 0.

least the operation of multiplication on floating point numbers that are at least 32 bits wide," in accordance with the claim language.

207. In addition, it is my opinion that a device satisfying this limitation was actually reduced to practice in this country, publicly known and used in this country, in public use in this country, and not abandoned, suppressed, or concealed. By way of example, as discussed above, to confirm and validate our conclusions regarding the maximum number of complete VFLOAT floating-point multiplication and addition operators that could fit on a single one of the three Xilinx Virtex XCV1000 FPGA on our WILDSTAR board, we synthesized and mapped that maximum number of operators to the FPGA. For the C2 format, for example, this means that we synthesized a design containing 61 C2 Multipliers and mapped that design to FPGA hardware (*i.e.*, to one of the three Xilinx Virtex XCV1000 FPGA on our WILDSTAR board), such that the FPGA contained circuitry embodying that many C2 Multipliers, as discussed above.

208. By way of further example, as discussed above, I personally traveled to LANL multiple times to discuss our work relating to VFLOAT with officials there, and ultimately delivered VFLOAT to LANL pursuant to the sub-contract discussed above (*i.e.*, "Acceleration of Scene Classification and Spectral Unmixing with Reconfigurable Computing," as identified in my CV). These discussions with LANL included, for example, presenting and describing various custom floating-point number formats that could be used with VFLOAT, as well as our synthesis results regarding the number of complete arithmetic operators that could be implemented within a single FPGA unit using various custom floating-point formats.

209. By way of additional example, as discussed above, during my presentation at HPEC 2002, I discussed and disclosed that our system setup used a WILDSTAR reconfigurable computing engine from Annapolis Micro and that the WILDSTAR board had three Xilinx Virtex XCV1000 FPGAs.[150] I also discussed and disclosed that we had developed VFLOAT in VHDL and mapped VFLOAT arithmetic modules to the Xilinx FPGAs.[151] I further discussed and

[150] *See* Exhibit C at 17 (HPEC 2002 Slides).
[151] *See* Exhibit C at 17 (HPEC 2002 Slides).

disclosed how to assemble VFLOAT modules to create a complete circuit for performing floating-point arithmetic operations—*i.e.*, using two *denorm* modules, one floating-point arithmetic module, and one *rnd_norm* module.[152] I also presented our synthesis results and conclusions regarding the number of complete floating-point arithmetic operators that would fit on the hardware we used, including that at least 50 complete floating-point multipliers built from VFLOAT modules and using a 16-bit floating-point format would fit on a single Xilinx Virtex XCV1000 FPGA.[153]

210. By way of still further example, as discussed above, the RPL lab itself, where our physical workstation was located, was a shared space in which many other students and faculty members of the Department of Electrical and Computer Engineering routinely worked and to which they had unrestricted access. In addition, as also explained above, many members of the Northeastern University community learned of and became familiar with VFLOAT during the time it was being developed and in subsequent years, including various graduate students as well as Laurie Smith King, a Professor of Computer Science at Holy Cross University who spent a sabbatical at Northeastern University.

211. Finally, as discussed above, at the MAPLD ’04 conference in September 2004, Xiaojun Wang and I presented an overview of our work related to VFLOAT, including our then-recent development and addition of modules for performing floating-point division (*fp_div*) and floating-point square root (*fp_sqrt*) to VFLOAT.[154]

212. Rather than enumerate here every public presentation, conference proceeding, or other public disclosure relating to VFLOAT that demonstrates how our work was publicly known and used in this country, in public use in this country, and not abandoned, suppressed, or concealed, I incorporate in its entirety my discussion above of the various public disclosures relating to VFLOAT, as well as the contents of those various public disclosures.

---

[152] *See* Exhibit C at 10 (HPEC 2002 Slides).
[153] *See* Exhibit C at 3, 18-19 (HPEC 2002 Slides).
[154] *See* Exhibit J at 1 (MAPLD ’04 Abstract).

213. For these reasons, it is my opinion that a device satisfying this limitation was actually reduced to practice in this country, publicly known and used in this country, in public use in this country, and not abandoned, suppressed, or concealed.

6. "wherein the number of LPHDR execution units in the device exceeds by at least one hundred the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide"

214. For the reasons explained below, it is my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was publicly known and/or used by others in this country before the priority date of the Asserted Patents.

215. For the reasons explained below, it is also my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was in public use in this country more than one year before the priority date of the Asserted Patents.

216. For the reasons explained below, it is further my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was made in this country by another inventor who had not abandoned, suppressed, or concealed the invention.

217. To begin with, I note that the language of this claim limitation very closely mirrors the previous limitation reciting "wherein the number of LPHDR execution units in the device exceeds the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide," with the sole exception that this claim limitation further requires that the "number of LPHDR execution units" exceeds by at least one hundred "the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide." Given the overlap in claim language, I incorporate by reference the opinions, analysis, and supporting facts addressed to that limitation as applicable here.

218. For example, as discussed above regarding the limitation reciting "at least one first low precision high-dynamic range (LPHDR) execution unit . . .", it is my opinion that a C2 Multiplier constitutes an "execution unit" as claimed (*i.e.*, "processing element comprising an arithmetic circuit paired with a memory circuit" in accordance with the Court's construction of "execution unit"[155]) and further satisfies the claim language regarding "low precision high-dynamic range (LPHDR)." By way of further example, as discussed above, to confirm and validate our conclusions regarding the maximum number of complete VFLOAT floating-point multiplication and addition operators that could fit on a single one of the three Xilinx Virtex XCV1000 FPGA on our WILDSTAR board, we synthesized and mapped that maximum number of operators to the FPGA. For the C2 format, for example, this means that we synthesized a design containing 61 C2 Multipliers and mapped that design to FPGA hardware (*i.e.*, to one of the three Xilinx Virtex XCV1000 FPGA on our WILDSTAR board), such that the FPGA contained circuitry embodying that many C2 Multipliers, as discussed above.

219. In addition, as described above, it is my opinion that the claimed "non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide" was at most four in view of the workstation that we used to implement C2 Multipliers. As discussed above in the section describing our system setup and implementation of VFLOAT at Northeastern University, our WILDSTAR board (on which we implemented C2 Multipliers) was installed in a standard Intel-based, x86 host workstation with a commercially available Intel Pentium III processor. As also discussed above, the Intel Pentium III processor contained four IEEE standard floating-point multipliers, which were each capable of performing multiplication on IEEE single-precision floating-point values (*i.e.*, floating-point numbers in the 32-bit IEEE single-precision format).

220. Assuming that each of the IEEE single-precision floating-point multipliers in the Pentium III is a "processing element comprising an arithmetic circuit paired with a memory

[155] *See* Dkt. 354, Mem. and Order on Claim Construction at 25.

circuit" in accordance with the Court's construction of "execution unit,"[156] it is my opinion that each of these multipliers is "adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide" in accordance with the claim language because they are designed to perform floating-point multiplication on 32-bit IEEE single-precision numbers, which are "floating point numbers that are at least 32 bits wide." Moreover, four is a "non-negative integer number" in accordance with the claim language. Thus, in my opinion, the "non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide," in accordance with the claim language, is at most four based on the number of IEEE single-precision floating-point multipliers in the Pentium III.[157]

a. Anticipation Based on the Lack of a New and Unexpected Result in this Limitation

221. As discussed above in the section of my report setting forth my understanding of relevant legal principles, it is my understanding that, in general, the mere duplication or repetition of parts or elements of a claimed invention may, in certain circumstances, be deemed to have no significance in the patentability of the claim over the prior art when no new and unexpected result is produced by the duplication or repetition. For example, I understand that merely increasing or decreasing the number or quantity of a specific component or element of a claimed invention, to yield a different number or quantity as compared to the prior art, will generally not be deemed a patentable distinction over the prior art unless the increase or decrease produces a new and unexpected result.

---

[156] *See* Dkt. 354, Mem. and Order on Claim Construction at 25.

[157] If the IEEE single-precision floating-point multipliers in the Pentium III are deemed not to constitute "execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide" under the Court's construction of "execution unit," then it is my alternative opinion that the maximum number of "execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide" in our workstation was zero, and my ultimate opinion regarding this claim limitation would remain unchanged, because 0 is a "non-negative integer" in accordance with the claim language and because the number 61 is greater than (*i.e.*, exceeds) the number 0.

222. I have been asked to consider and offer my opinion on whether any new and unexpected results arise from the "exceeds by at least one hundred" requirement in this claim limitation—*i.e.*, the requirement that "the number of LPHDR execution units in the device" exceeds "by at least one hundred" the "non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide." In my opinion, the "exceeds by at least one hundred" requirement in this claim limitation constitutes the mere duplication or repetition of elements of the claimed invention, and that no new and unexpected result is produced by such duplication or repetition.

223. As already noted, the language of this claim limitation very closely mirrors the previous limitation reciting "wherein the number of LPHDR execution units in the device exceeds the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide," with the sole exception that this claim limitation further requires a greater minimum number of "LPHDR execution units." In particular, whereas the prior claim limitation only requires that the number of "LPHDR execution units in the device" be at least one more than (i.e., that it "exceed[]") the "non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide," this claim limitation requires that there be at least at least 100 more of the former type of execution units ("LPHDR execution units") than the latter type of execution units ("execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide") in the claimed "device." Similarly, the first limitation of this claim—the limitation requiring that the claimed "device" "compris[e] at least one first low precision high-dynamic range (LPHDR) execution unit adapted to execute a first operation . . ."—already requires by its express terms "at least one first low precision high-dynamic range (LPHDR) execution unit." Thus, in my opinion, the "exceeds by at least one hundred" requirement in this claim limitation clearly calls for repeating or duplicating one particular element of the claimed invention—namely, repeating or duplicating the "LPHDR execution

units" (*e.g.*, at least 99 more instances of an "LPHDR execution unit[]" in the claimed "device" than already required by the prior limitations of the claim).

224. Notably, the claim recites no particular details on the configuration of the additional instances of an "LPHDR execution unit[]" required in the claimed "device" by this claim limitation, nor does the claim require any differences between the at least one LPHDR execution unit already required due to the prior limitations and the additional "LPHDR execution units" required by this claim limitation; all the additional "LPHDR execution units" required by this claim limitation can be identical to each other and to the LPHDR execution units already required by the prior claim limitation. In other words, the "exceeds by at least one hundred" requirement in this claim limitation can be satisfied by the presence of *n* identical "LPHDR execution units" in the claimed "device," so long as (*n* - 100) is greater than or equal to "the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide."

225. In my further opinion, the particular repetition or duplication of the recited "LPHDR execution units" required by this limitation (such that there must be at least 99 more "LPHDR execution units" in the claimed "device" than are already required by prior limitations) leads to no new and unexpected result as compared to some smaller (or larger) number of "LPHDR execution units" in the claimed "device." To begin with, as I've already noted, the prior limitations in the claim require at least one "LPHDR execution unit." For example, as I've noted, the first limitation of this claim—the limitation requiring that the claimed "device" "compris[e] at least one first low precision high-dynamic range (LPHDR) execution unit adapted to execute a first operation . . ."—already requires by its express language "at least one first low precision high-dynamic range (LPHDR) execution unit." It is my opinion that a person of ordinary skill in the art would have understood that the extent of repetition or duplication of the recited "LPHDR execution units" called for by the "exceeds by at least one hundred" requirement does not lead to any new and unexpected result.

226. As an example to illustrate this point, consider a "device" in accordance with the claim language that comprises 5 of the claimed "execution units . . . adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide" and otherwise meets the claim, but only comprises 55 of the claimed "LPHDR execution units," such that it did not satisfy the "exceeds by at least one hundred" requirement in this claim limitation. The addition of another 55 of the claimed "LPHDR execution units" through simple repetition or duplication—for a total of 110, which would then satisfy the "exceeds by at least one hundred" requirement in this claim limitation—leads to no new and unexpected result. The principal result is to potentially increase the capacity of the claimed "device" to execute multiple operations in parallel, but that result is neither new nor unexpected. Rather, it is a wholly expected result and one that is decidedly not new because parallelism existed even before the repetition or duplication of another 55 of the claimed "LPHDR execution units" (to reach 110) and a person of ordinary skill in the art would have fully expected that the result of increasing the number of execution units would be, at most, a correspondingly linear increase in the capacity to execute operations in parallel. The same is true of simple repetition or duplication of the claimed "LPHDR execution units" to increase the number of such units in the claimed "device" from, say, 50 to 110, 45 to 110, 40 to 110, and so on.

227. Although the shared specification of the Asserted Patents claims that "it is commonly believed by those having ordinary skill in the art, that . . . massive amounts of LPHDR computation . . . is not practical as a substrate for moderately general computing,"[158] this does not, in my opinion, demonstrate that the extent of repetition or duplication of the recited "LPHDR execution units" called for by the "exceeds by at least one hundred" requirement leads to any new and unexpected result, and in any event I disagree with the proposition that persons of ordinary skill in the art viewed large amounts of low-precision, high dynamic range computation as impractical or not useful. For instance, notwithstanding the assertion quoted

[158] *E.g.*, '273 patent at 6:58-63.

above, the shared specification of the Asserted Patents ascribes no significance to the particular repetition or duplication of the recited "LPHDR execution units" called for by the "exceeds by at least one hundred" requirement, either based on the absolute number of LPHDR units or their number in relation to the number of "execution units . . . adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide." The specification does not say, for example, that "exceed[ing] by at least one hundred" leads to any unique or unexpected result as compared to, say, "exceed[ing] by at least" 90, 50, 40, or even 10. Indeed, the specification does not even specifically mention or discuss the "exceeds by at least one hundred" requirement. For instance, while the specification offers illustrative examples of different degrees of repetition or duplication of "LPHDR arithmetic elements," such as that "the number of LPHDR arithmetic elements in the processor or other device in certain embodiments of the present invention significantly exceeds (e.g., by at least 20 more than three times) the number of arithmetic elements in the processor or other device which are designed to perform high dynamic range arithmetic of traditional precision (such as 32 bit or 64 bit floating point arithmetic),"[159] it does not ascribe any new and unexpected result to the particular repetition or duplication of the recited "LPHDR execution units" called for by the "exceeds by at least one hundred" requirement in this limitation. Indeed, the specification offers a laundry list of different degrees of repetition or duplication of LPHDR arithmetic elements without ascribing a particular new and unexpected result to any of them:

> For certain devices (such as computers or processors or other devices) embodied according the present invention, the number of LPHDR arithmetic elements in the device (e.g., computer or processor or other device) exceeds the number, possibly zero, of arithmetic elements in the device which are designed to perform high dynamic range arithmetic of traditional precision (that is, floating point arithmetic with a word length of 32 or more bits). If NL is the total number of LPHDR elements in such a device, and NH is the total number of elements in the device which are designed to perform high dynamic range arithmetic of traditional precision, then NL exceeds T(NH), where TO is some function.

[159] *E.g.*, '273 patent at 2:20-27.

> ***Any of a variety of functions may be used as the function TO***. For example, in certain embodiments, T(NH) may be twenty plus three times NH, and the number of LPHDR arithmetic elements in the device may ***exceed twenty more than three times the number*** of arithmetic elements in the device, if any, designed to perform high dynamic range arithmetic of traditional precision. As another example, in certain embodiments, the number of LPHDR arithmetic elements in the device ***may exceed fifty more than five times the number*** of arithmetic elements in the device, if any, designed to perform high dynamic range arithmetic of traditional precision. As yet another example, in certain embodiments, the number of LPHDR arithmetic elements in the device ***may exceed one hundred more than five times the number*** of arithmetic elements in the device, if any, designed to perform high dynamic range arithmetic of traditional precision. As yet another example, in certain embodiments, the number of LPHDR arithmetic elements in the device ***may exceed one thousand more than five times the number*** of arithmetic elements in the device, if any, designed to perform high dynamic range arithmetic of traditional precision. As yet another example, in certain embodiments, the number of LPHDR arithmetic elements in the device ***may exceed five thousand more than five times the number*** of arithmetic elements in the device, if any, designed to perform high dynamic range arithmetic of traditional precision.[160]

228. In my opinion, this list only serves, if anything, to highlight the arbitrary nature of the "exceeds by at least one hundred" claim requirement. Notably, that claim requirement is not even among those listed in the specification; rather, it seems to conflict with the inclusion of a different configuration in the list excerpted above: "the number of LPHDR arithmetic elements in the device may exceed ***one hundred more than five times*** the number of arithmetic elements in the device."[161]

229. Moreover, the specification's claim appears to be that the new and unexpected result in the claimed invention, if any, arises from the use of LPHDR computation, not from any particular degree of repetition or duplication of the recited "LPHDR execution units": "it is commonly believed by those having ordinary skill in the art, that LPHDR computation . . .

[160] *E.g.*, '273 patent at 27:63-28:32 (emphases added).
[161] *E.g.*, '273 patent at 27:63-28:32 (emphasis added).

whether performed in a massively parallel way or not, is not practical as a substrate for moderately general computing"[162]

230. In any event, as noted above, I disagree with the proposition that persons of ordinary skill in the art viewed low-precision, high dynamic range computation, or large amounts of such computation, as impractical or not useful. Indeed, I have spent much of my career focused on applications of highly parallel computation of low precision and high dynamic range. In fact, the very application for which we originally developed VFLOAT involved massively parallel computation on low-precision and high dynamic range datasets, namely multi-spectral and hyper-spectral satellite imagery for LANL. Moreover, I can think of—and reserve the right to specifically identify, if requested—many examples of individuals and organizations interested in and implementing parallel computation of low precision and high dynamic range before the priority date of the Asserted Patents.

231. For these reasons, it is my opinion that the particular repetition or duplication of the recited "LPHDR execution units" required by this limitation (such that there must be at least 99 more "LPHDR execution units" in the claimed "device" than are already required by prior limitations) leads to no new and unexpected result.

232. For the reasons discussed above, it is my opinion that the particular repetition or duplication of the recited "LPHDR execution units" required by this claim limitation (*i.e.*, by the "exceeds by at least one hundred" requirement) may be deemed to have no significance in the patentability of the claim over the prior art, and that our implementation of 61 C2 Multipliers in FPGA hardware therefore anticipates the claim.

[162] *E.g.*, '273 patent at 6:58-62 (emphasis added).

b. Anticipation and/or Obviousness Based on Making, Using, and Disclosing the VFLOAT System, Where the Lower-Precision Multipliers Exceeded By At Least One Hundred the Single-Precision Multipliers

233. To the extent that the "exceeds by at least one hundred" requirement in this claim limitation is deemed to qualify as a patentably distinct limitation rather than mere duplication or repetition of an element, it is my opinion that the VFLOAT system we made and publicly used and disclosed met and/or rendered obvious this claim limitation.

234. As discussed above, the VFLOAT system that we built used a WILDSTAR board with three Xilinx Virtex XCV1000 FPGAs. Furthermore, we used the BSP from Annapolis Micro to program one of those FPGAs to have 61 C2 Multipliers. That VFLOAT system could use the same BSP to program the two other identical Xilinx FPGAs on the WILDSTAR board to each have 61 C2 Multipliers—for a total of 183 C2 Multipliers across the three Xilinx FPGAs on the WILDSTAR board—as any person of ordinary skill would have understood. Thus, the VFLOAT system that we built was tested sufficiently to show that it will work for its intended purpose: putting numerous, parallel, complete multiplier circuits using a floating-point format with 1 sign bit, 6 exponent bits, and 9 fraction bits onto multiple FPGAs on a WILDSTAR board. That VFLOAT system that we built and tested would have worked to include 183 C2 Multipliers, which exceeds by at least one hundred the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide (four). Thus, it is my opinion that the VFLOAT system that we built and programmed with C2 Multipliers meets this element.

235. It is further my opinion that public disclosures regarding the VFLOAT system we built and tested meets this element. Specifically, as detailed above, both Mr. Belanovic and I gave public presentations explaining the use of a WILDSTAR board with three Xilinx Virtex XCV1000 FPGAs programmed to implement lower-precision floating-point formats (including the C2 format with 1 sign bit, 6 exponent bits, and 9 fraction bits). These presentations explained the benefits of additional parallelism and the fact that we could load at least one

hundred more lower-precision multiplier circuits as compared to the non-negative integer number of execution units in the system. In particular, Mr. Belanovic disclosed (among others) the C2 format with 1 sign bit, 6 exponent bits, and 9 fraction bits at his thesis defense; we disclosed that and other such lower-precision formats in our meetings with LANL; and I disclosed at HPEC that over 50 multipliers using a lower-precision, 16-bit format would fit onto a single one of the three FPGAs on the WILDSTAR board.

236. For these reasons, it is my opinion that a device satisfying this limitation was actually reduced to practice in this country, publicly known and used in this country, in public use in this country, and not abandoned, suppressed, or concealed.

237. To the extent that the VFLOAT system that we made and publicly used and disclosed does not meet this element because it contained 61 C2 Multipliers on a single one of the three FPGAs on the WILDSTAR board, it would have been obvious to one of skill in the art that the other two FPGAs could each have been programmed to also contain 61 C2 Multipliers. A person of ordinary skill in the art would have understood that there were a number of useful and practical applications of implementing a large number of parallel floating-point multipliers, and therefore would have been motivated to implement the maximum number of such units that they could within a given FPGA or set of multiple FPGAs within a single reconfigurable computing engine such as the WILDSTAR board that we used. Indeed, the public presentations related to the VFLOAT system emphasized the benefits of increased parallelism and would have motivated one of skill in the art to program the other two FPGAs on the WILDSTAR board to also implement those multipliers. Furthermore, this would have been an obvious modification of the VFLOAT system because the BSP provided by Annapolis Micro allowed one to easily program the other two FPGAS on the WILDSTAR board to implement the same layout of C2 Multipliers that we used to program the one FPGA as part of our research and testing. As detailed above, our public disclosures specifically referred to the WILDSTAR board and its three FPGAs, and one of skill in the art using such a board would have used the BSP provided by Annapolis Micro to program the board.

238. Additionally, it is also my opinion that there were additional useful configurations of the Wildstar board involving the construction of C2 Multipliers on the Xilinx Virtex XCV1000 FPGAs that would result in the VFLOAT system satisfying this limitation and that a person of ordinary skill in the art would have been motivated to implement. For instance, matrix multiplication is a highly useful and ubiquitous mathematical operation that can benefit from the implementation of a large number of multiplication units that can operate in parallel. For example, it is a mathematical fact—and a person of ordinary skill in the art would have understood—that multiplying two n x n matrices (or multiplying an m x n matrix with an n x p matrix) requires the calculation of what is known as a "dot product." Roughly speaking, a dot product in this context is the "product" of one "row" of the first matrix with one "column" of the second matrix. Dot product computation requires multiplying n pairs of numbers together and performing n - 1 additions (*i.e.*, summing together the products of the n multiplications), and the multiplications can be performed in parallel, as a person of ordinary skill in the art would have understood. Thus, a person of ordinary skill in the art would have understood that dot product computation is a practical and highly useful application of implementing a large number of floating-point multiplication operators that can operate in parallel. Furthermore, a person of ordinary skill in the art would have understood that, even assuming the implementation of n floating-point multiplication operators and n - 1 floating-point addition operators in the same FPGA (*e.g.*, to perform all additions and multiplications for a single dot product computation on one FPGA), a total of 105 C2 Multipliers would still fit on the three XCV1000 FPGAs on a single WILDSTAR board, based on simple arithmetic to account for the relative size of floating-point multiplication and addition operators designed for the C2 format. Moreover, a person of ordinary skill in the art would have understood that an even fewer number of floating-point addition operators (*e.g.*, n/2 rather than n - 1) could support dot product computation (*e.g.*, if some adders are used multiple times in the computation), which would allow an even greater number of C2 Multipliers (*i.e.*, greater than 105) to fit on the three XCV1000 FPGAs on a single WILDSTAR board. It is my opinion that an implementation of at least105 C2 Multipliers

satisfies the "exceeds by at least one hundred" requirement, since, as I noted above, the system had no more than 4 execution units adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide.

c. Obviousness in View of 2008-Era FPGA Technology

239. Even if our system containing 61 C2 Multipliers is considered or deemed not to anticipate the claim with respect to this limitation, it is my opinion that, as of 2008, it would have been obvious to implement C2 Multipliers in sufficient quantity that it would have satisfied this claim limitation.

240. As discussed above in the section of my report providing technical background regarding FPGAs, FPGA technology advanced considerably from the time in approximately 2002 that we first developed VFLOAT and implemented C2 Multipliers on the Xilinx Virtex XCV1000 FPGAs on our WILDSTAR board, to the priority date of the Asserted Patents. For example, as discussed above, the Xilinx Virtex XCV1000 FPGA (of which there were three on our WILDSTAR board) had the equivalent of just over 1,000,000 (1,124,022) system gates. As also discussed above, by late 2006, when the Xilinx Virtex-5 family of FPGAs was commercially available, FPGA technology had advanced considerably. For example, the Xilinx Virtex-5 XC5VLX330T had approximately 9.4 million gate equivalents—a more than eight-fold increase as compared to the Xilinx Virtex XCV1000 FPGAs that were on our WILDSTAR board.

241. In my opinion, a person of ordinary skill in the art would have understood that the number of a given circuit that can be implemented on an FPGA grows roughly linearly with the increase in the gate equivalents of the FPGA. For example, an FPGA with $2n$ gate equivalents can accommodate roughly twice as many of a given circuit as a similar FPGA with only n gate equivalents, and in my opinion a person of ordinary skill in the art would have understood this.

242. Thus, by the time of the priority date of the Asserted Patents, a person of ordinary skill in the art would have understood that FPGAs were available that had at least eight times as

many gate equivalents as the Xilinx Virtex XCV1000 FPGA that we used (such as the Xilinx Virtex-5 XC5VLX330T), and would have further understood that such FPGAs would accommodate at least eight times as many C2 Multipliers as the Virtex XCV1000. Thus, given that we were able to implement 61 C2 Multipliers on the Virtex XCV1000, a person of ordinary skill in the art as of the priority date of the Asserted Patents would have understood that a single 2008-era FPGA, such as the XC5VLX330T, would accommodate well in excess of 400 C2 Multipliers (*e.g.*, at least approximately 488 based on the at least eight-fold increase in gate equivalents). Moreover, in my opinion, a person of ordinary skill in the art would have been motivated to extend our work using then-current FPGA technology (*i.e.*, as of the priority date of the Asserted Patents), such as a Xilinx Virtex-5 FPGA, in order to reap the benefits of advances in FPGA technology such as the increased density of then-current FPGAs and would have been further motivated to maximize the number of C2 Multipliers on such FPGAs because of the useful and practical applications for implementing a large number of parallel floating-point multipliers, as discussed above. Furthermore, as discussed above, a person of ordinary skill in the art would have understood that there were additional useful configurations of FPGAs that a POSA would have been motivated to implement, such as a configuration for performing matrix multiplication. Even assuming the implementation of a large number of floating-point addition operators on the same FPGA (*e.g.*, $n/2$ or even n -1 adders, to perform addition on the products of multiplying the n pairs of matrix elements in a dot product computation), a person of ordinary skill in the art would have understood that at least 244 C2 Multipliers would fit on a single 2008-era FPGA (such as the Xilinx Virtex-5 XC5VLX330T), due to the relative size of floating-point addition and multiplication operators built for the C2 format using VFLOAT.

7. "wherein the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/1,000,000 through 1,000,000"

243. For the reasons explained below, it is my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was publicly known and/or used by others in this country before the priority date of the Asserted Patents.

244. For the reasons explained below, it is also my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was in public use in this country more than one year before the priority date of the Asserted Patents.

245. For the reasons explained below, it is further my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was made in this country by another inventor who had not abandoned, suppressed, or concealed the invention.

246. My opinions regarding this limitation, and the facts and reasons supporting those opinions, are essentially the same as for the previously addressed limitation reciting that "the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/65,000 through 65,000." I therefore incorporate by reference the opinions, analysis, and supporting facts (including the relevant math) addressed to that limitation as applicable here, and reach the same conclusions and hold the same opinions regarding this claim limitation as for that previously addressed limitation.

247. By way of example, as discussed above, the inputs to a C2 Multiplier are floating-point numbers in the C2 format (which, as discussed previously, has 1 sign bit, a 6-bit biased exponent, and 9 fraction bits). As also discussed above, a biased exponent that is 6 bits (such as in the C2 format) can range in value from at least -30 to 31,[163] such that the resulting floating-point number has a dynamic range at least as wide as from 2^{-30} (which is 1/1,073,741,824) on the low end to over 2^{31} (which is 2,147,483,648) on the high end. In my opinion, the possible valid inputs to the C2 Multiplier have such a dynamic range because both numerical inputs to the C2 Multiplier use the C2 floating-point format. In my opinion, such a dynamic range is "at least as wide as from 1/1,000,000 through 1,000,000" in accordance with the claim language.

[163] As previously noted and also discussed above in the section of my report providing technical background regarding floating-point numbers, floating-point formats that follow IEEE convention reserve the smallest and largest exponents (all zeros or all ones) to represent special values.

248. For these reasons, it is my opinion that a device satisfying this limitation was actually reduced to practice in this country, publicly known and used in this country, in public use in this country, and not abandoned, suppressed, or concealed.

B. '156 Patent, Claim 7

249. Claim 7 of the '156 patent depends from claim 3, which in turn depends from claim 2, which in turn depends from independent claim 1. Including the limitations from claims 1, 2, and 3, claim 7 reads:

> **1.** A device comprising:
>
> at least one first low precision high dynamic range (LPHDR) execution unit adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value,
>
> wherein the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/65,000 through 65,000 and for at least X=5% of the possible valid inputs to the first operation, the statistical mean, over repeated execution of the first operation on each specific input from the at least X% of the possible valid inputs to the first operation, of the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input differs by at least Y=0.05% from the result of an exact mathematical calculation of the first operation on the numerical values of that same input; and
>
> at least one first computing device adapted to control the operation of the at least one first LPHDR execution unit.
>
> **2.** The device[164] of claim 1, wherein the at least one first computing device comprises at least one of a central processing unit (CPU), a graphics processing unit (GPU), a field programmable gate array (FPGA), a microcode-based processor, a hardware sequencer, and a state machine.
>
> **3.** The device of claim 2, wherein the number of LPHDR execution units in the device exceeds by at least one hundred the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide.

[164] I understand that a Certificate of Correction was issued for the '156 patent, which corrected the preamble of claim 2 from "The method of claim 1" to "The device of claim 1."

> **7.** The device of claim 3, wherein the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/1,000,000 through 1,000,000.

250. For the reasons explained below, it is my opinion that claim 7 of the '156 patent is invalid as anticipated due to the claimed invention being known or used by others in this country before the priority date of the Asserted Patents, being in public use in this country more than one year prior to the priority date for the Asserted Patents, and/or being previously made in this country by another inventor who had not abandoned, suppressed, or concealed the invention, based on the creation, public availability, public disclosure, public knowledge, and public use of and regarding VFLOAT and its use on FPGA hardware. In addition, it is my further opinion that, even if claim 7 of the '156 patent is not anticipated, the claimed invention would have been obvious to a person having ordinary skill in the art at the time based on the creation, public availability, public disclosure, public knowledge, and public use of and regarding VFLOAT in combination with FPGA hardware available at the time of the claimed invention.

1. "A device"

251. For this element of claim 7 of the '156 patent, I incorporate my analysis and opinions for the corresponding element of claim 56 of the '273 patent. *See* § VII.A.1.

2. "comprising at least one first low precision high dynamic range (LPHDR) execution unit adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value"

252. For this element of claim 7 of the '156 patent, I incorporate my analysis and opinions for the corresponding element of claim 56 of the '273 patent. *See* § VII.A.2.

3. "wherein the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/65,000 through 65,000"

253. For this element of claim 7 of the '156 patent, I incorporate my analysis and opinions for the corresponding element of claim 56 of the '273 patent. *See* § VII.A.3.

4. "for at least X=5% of the possible valid inputs to the first operation, the statistical mean, over repeated execution of the first operation on each specific input from the at least X% of the possible valid inputs to the first operation, of the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input differs by at least Y=0.05% from the result of an exact mathematical calculation of the first operation on the numerical values of that same input"

254. For this element of claim 7 of the '156 patent, I incorporate my analysis and opinions for the corresponding element of claim 56 of the '273 patent. *See* § VII.A.4.

5. "at least one first computing device adapted to control the operation of the at least one first LPHDR execution unit"

255. For the reasons explained below, it is my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was publicly known and/or used by others in this country before the priority date of the Asserted Patents.

256. For the reasons explained below, it is also my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was in public use in this country more than one year before the priority date of the Asserted Patents.

257. For the reasons explained below, it is further my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was made in this country by another inventor who had not abandoned, suppressed, or concealed the invention.

258. As discussed above in the section describing our system setup and implementation of VFLOAT at Northeastern University, our WILDSTAR board (on which we implemented C2 Multipliers) was installed in a standard Intel-based, x86 host workstation with a commercially available Intel Pentium III processor. It is my opinion that our workstation constitutes a "first computing device" in accordance with the claim language, and that a person of ordinary skill in the art would have understood that such a workstation constitutes a "first computing device" in accordance with the claim language, because such a workstation is a device that performs computation and is used to carry out computing tasks.

259. In addition, in my opinion, our workstation was "adapted to control the operation of the at least one first LPHDR execution unit" in accordance with the claim language. I understand that the use of "the" in the claim language reciting "the at least one first LPHDR execution unit" refers back to the previously-recited "at least one first low precision high dynamic range (LPHDR) execution unit" that appears in the first limitation of this claim. As discussed above, it is my opinion that each of the C2 Multipliers we implemented in FPGA hardware satisfies the claim language regarding "low precision high-dynamic range (LPHDR) execution unit," and in particular the Court's construction of that claim language, and thus constitute "the at least one first LPHDR execution unit." In my opinion, our workstation was "adapted to control the operation of" these C2 Multipliers, in accordance with the claim language, both because the workstation is responsible for supplying data that the C2 Multipliers would operate on, as well as because it was responsible for initiating the operation of the FPGAs on which the C2 Multipliers were implemented. As discussed above, for example, our host workstation was adapted to send data to the memory on the WILDSTAR board, which could then be input directly into circuits implemented on the WILDSTAR board's FPGAs. In addition, as also discussed above, our workstation would initiate the operation of the circuitry on the FPGAs that implemented C2 Multipliers by sending a "start" signal to the WILDSTAR board. In my opinion, a person of ordinary skill in the art would have understood that each of these mechanisms is a way that our workstation was "adapted to control the operation of" the C2 Multipliers, in accordance with the claim language. Thus, for these reasons, it is my opinion that our workstation was "adapted to control the operation of the at least one first LPHDR execution unit" in accordance with the claim language.

260. For these reasons, it is my opinion that a device satisfying this limitation was actually reduced to practice in this country, publicly known and used in this country, in public use in this country, and not abandoned, suppressed, or concealed.

6. "wherein the at least one first computing device comprises at least one of a central processing unit (CPU), a graphics processing unit (GPU), a field programmable gate array (FPGA), a microcode-based processor, a hardware sequencer, and a state machine"

261. For the reasons explained below, it is my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was publicly known and/or used by others in this country before the priority date of the Asserted Patents.

262. For the reasons explained below, it is also my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was in public use in this country more than one year before the priority date of the Asserted Patents.

263. For the reasons explained below, it is further my opinion that a device, as recited in the context of the claimed invention, that meets this claim limitation was made in this country by another inventor who had not abandoned, suppressed, or concealed the invention.

264. I understand that the use of "the" in the language of this claim limitation reciting "the at least one first computing device" refers back to the previously-recited "at least one first computing device" that appears in the previous limitation of this claim. As discussed above regarding that previous claim limitation, it is my opinion that our workstation constitutes a "first computing device" in accordance with the claim language, and that a person of ordinary skill in the art would have understood that such a workstation constitutes a "first computing device" in accordance with the claim language. For example, as discussed above, our WILDSTAR board (on which we implemented C2 Multipliers) was installed in a standard Intel-based, x86 host workstation with a commercially available Intel Pentium III processor.

265. In my opinion, our workstation "comprises at least one of a central processing unit (CPU), a graphics processing unit (GPU), a field programmable gate array (FPGA), a microcode-based processor, a hardware sequencer, and a state machine," in accordance with the language of this claim limitation, for two independent reasons. First, as discussed above in the section of my report describing our system setup at Northeastern University, our workstation's CPU was a commercially available Intel Pentium III processor. In my opinion, a person of

ordinary skill in the art would have understood that an Intel Pentium III processor constitutes a "central processing unit (CPU)." In addition, as discussed several times throughout my report, the WILDSTAR board contained three Xilinx Virtex XCV1000 FPGAs. In my opinion, a person of ordinary skill in the art would have understood that an FPGA, such as a Xilinx Virtex XCV1000, constitutes a "a field programmable gate array (FPGA)" in accordance with the claim language. Thus, for these reasons, it is my opinion that our workstation was "comprises at least one of a central processing unit (CPU), a graphics processing unit (GPU), a field programmable gate array (FPGA), a microcode-based processor, a hardware sequencer, and a state machine" in accordance with the claim language.

266. For these reasons, it is my opinion that a device satisfying this limitation was actually reduced to practice in this country, publicly known and used in this country, in public use in this country, and not abandoned, suppressed, or concealed.

7. "wherein the number of LPHDR execution units in the device exceeds by at least one hundred the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide"

267. For this element of claim 7 of the '156 patent, I incorporate my analysis and opinions for the corresponding element of claim 56 of the '273 patent. *See* § VII.A.5.

8. "wherein the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/1,000,000 through 1,000,000"

268. For this element of claim 7 of the '156 patent, I incorporate my analysis and opinions for the corresponding element of claim 56 of the '273 patent. *See* § VII.A.6.